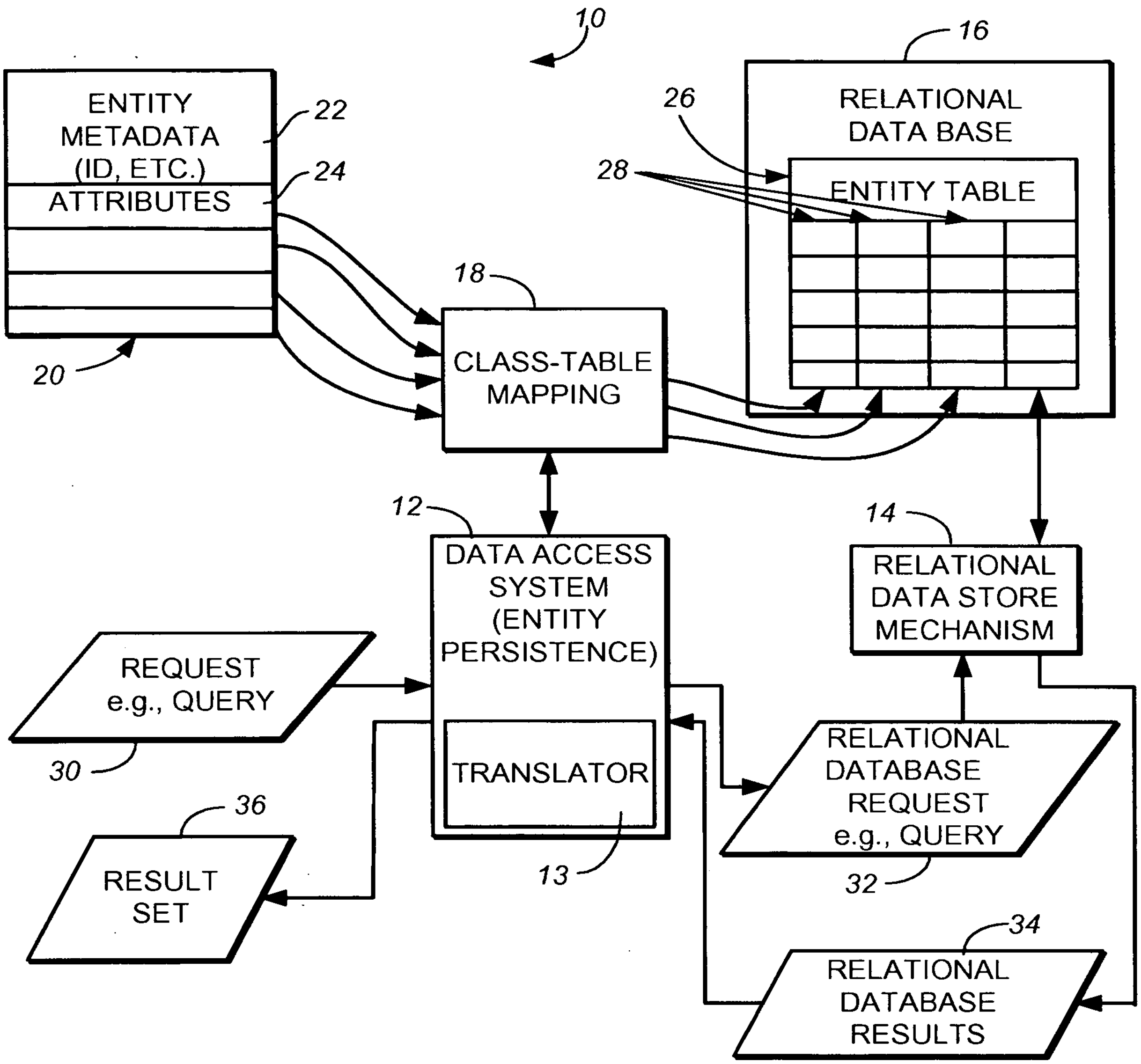

10
ENTITY METADATA (ID, ETC.)
22
ATTRIBUTES
24
20
26
28
16
RELATIONAL DATA BASE
ENTITY TABLE
18
CLASS-TABLE MAPPING
12
DATA ACCESS SYSTEM (ENTITY PERSISTENCE)
TRANSLATOR
13
14
RELATIONAL DATA STORE MECHANISM
REQUEST e.g., QUERY
30
36
RESULT SET
RELATIONAL DATABASE REQUEST e.g., QUERY
32
34
RELATIONAL DATABASE RESULTS

FIG._1

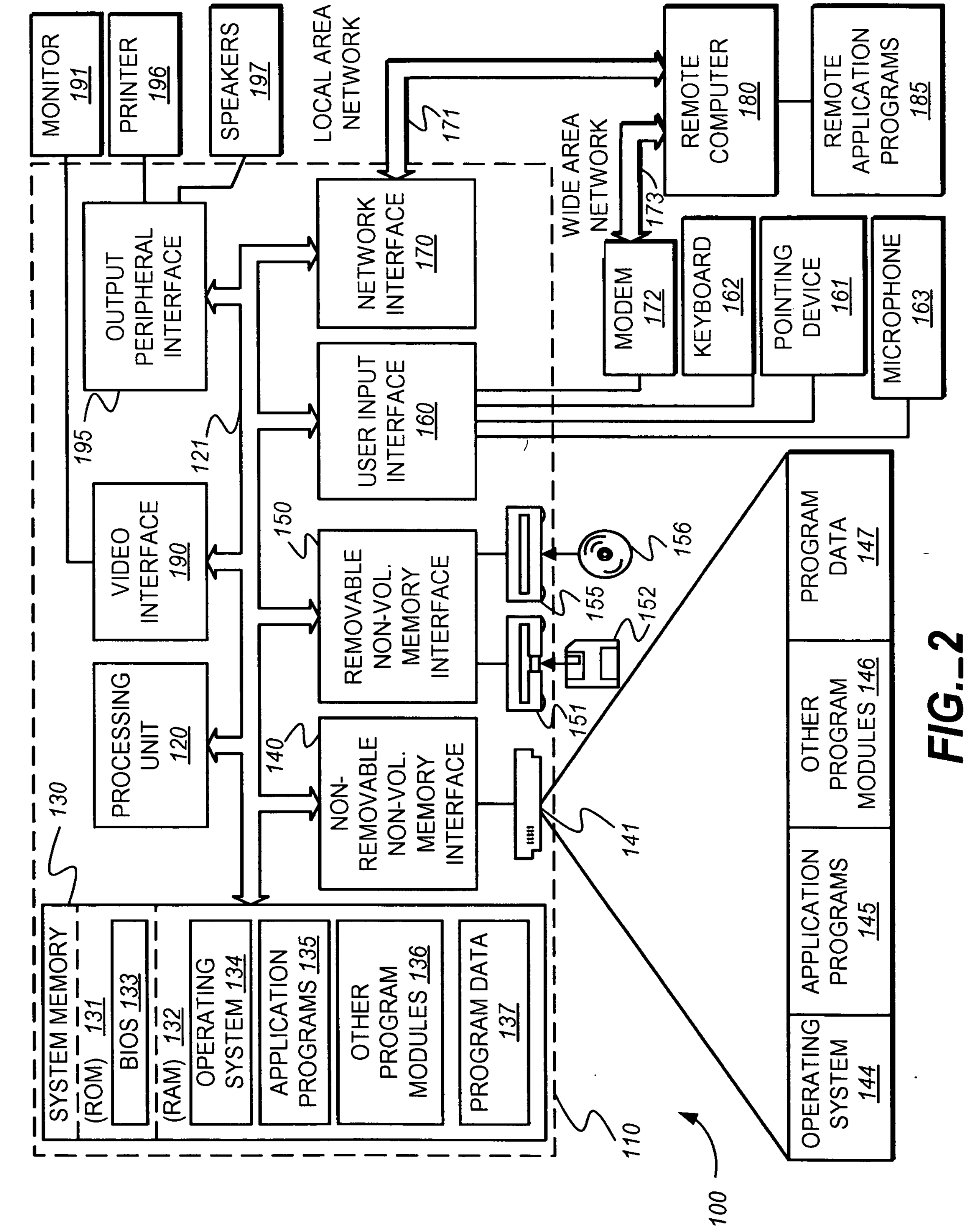

SYSTEM MEMORY
130
(ROM) 131
BIOS 133
(RAM) 132
OPERATING SYSTEM 134
APPLICATION PROGRAMS 135
OTHER PROGRAM MODULES 136
PROGRAM DATA 137
PROCESSING UNIT 120
VIDEO INTERFACE 190
OUTPUT PERIPHERAL INTERFACE
195
121
MONITOR 191
PRINTER 196
SPEAKERS 197
NON-REMOVABLE NON-VOL. MEMORY INTERFACE
140
REMOVABLE NON-VOL. MEMORY INTERFACE
150
USER INPUT INTERFACE 160
NETWORK INTERFACE 170
LOCAL AREA NETWORK
171
WIDE AREA NETWORK
173
MODEM 172
KEYBOARD 162
POINTING DEVICE 161
MICROPHONE 163
REMOTE COMPUTER 180
REMOTE APPLICATION PROGRAMS 185
141
151
152
155
156
OPERATING SYSTEM 144
APPLICATION PROGRAMS 145
OTHER PROGRAM MODULES 146
PROGRAM DATA 147
110
100

FIG._2

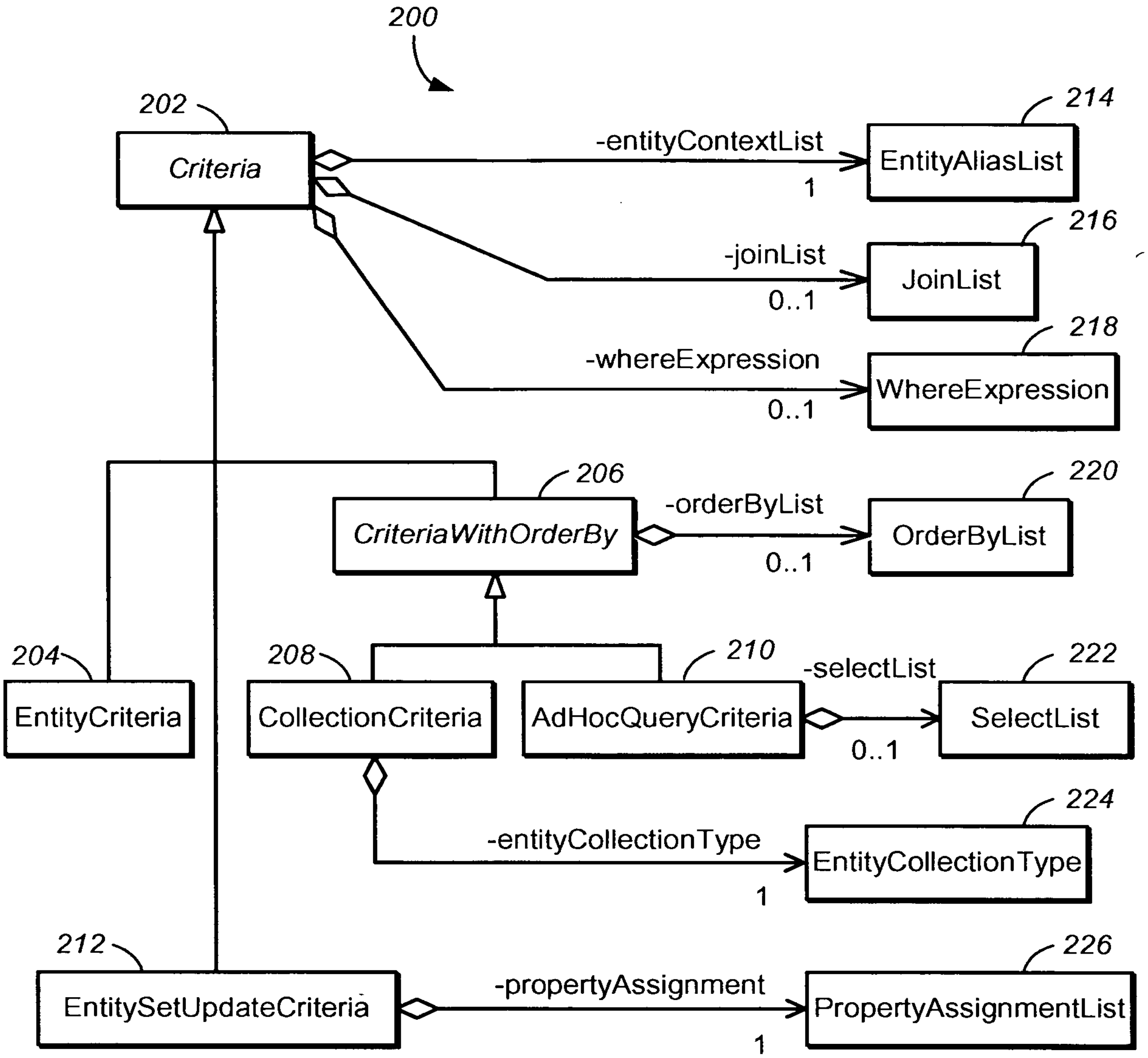

200
202
Criteria
-entityContextList
1
214
EntityAliasList
-joinList
0..1
216
JoinList
-whereExpression
0..1
218
WhereExpression
206
CriteriaWithOrderBy
-orderByList
0..1
220
OrderByList
204
EntityCriteria
208
CollectionCriteria
210
AdHocQueryCriteria
-selectList
0..1
222
SelectList
-entityCollectionType
1
224
EntityCollectionType
212
EntitySetUpdateCriteria
-propertyAssignment
1
226
PropertyAssignmentList

FIG._3

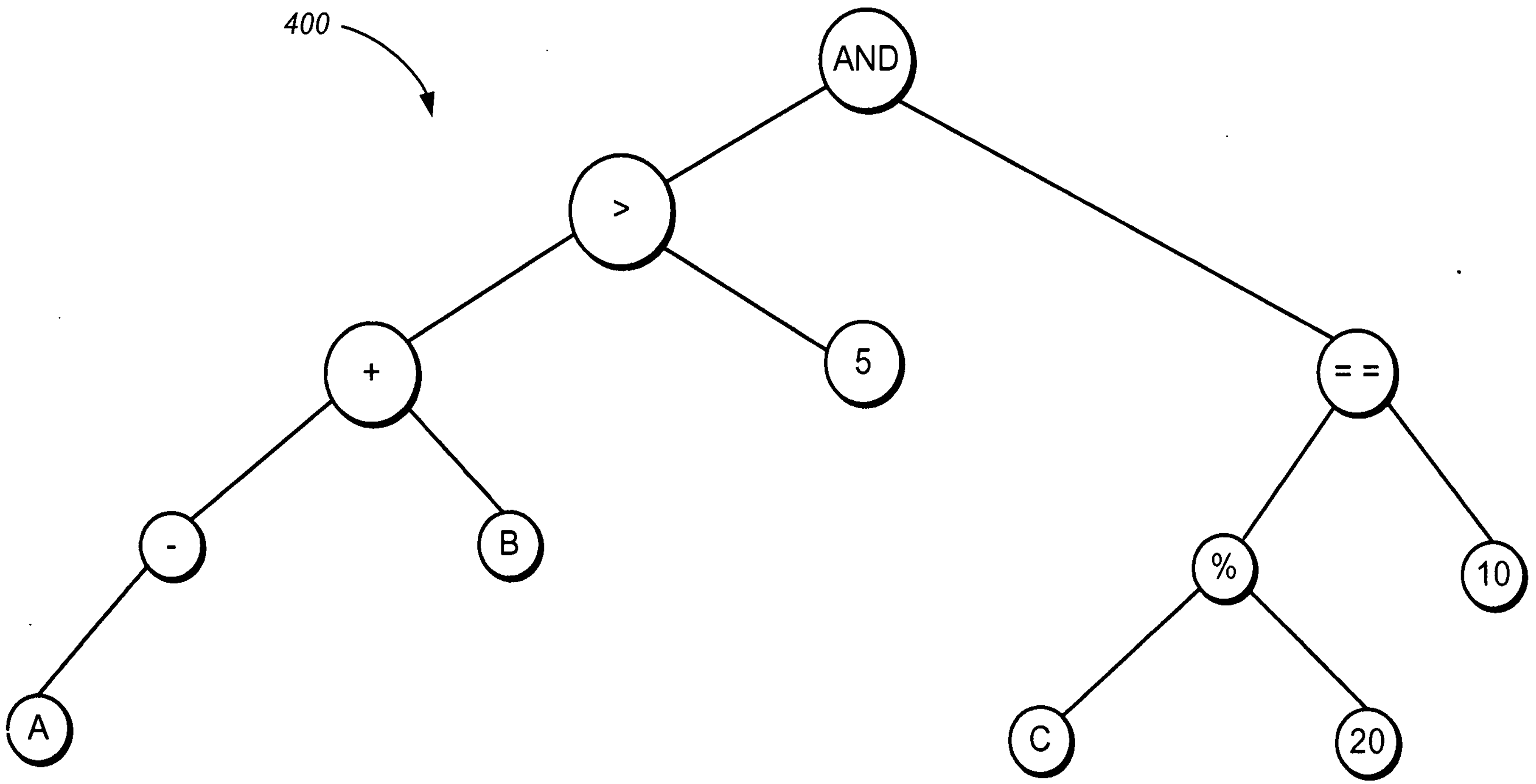


FIG._ 4A

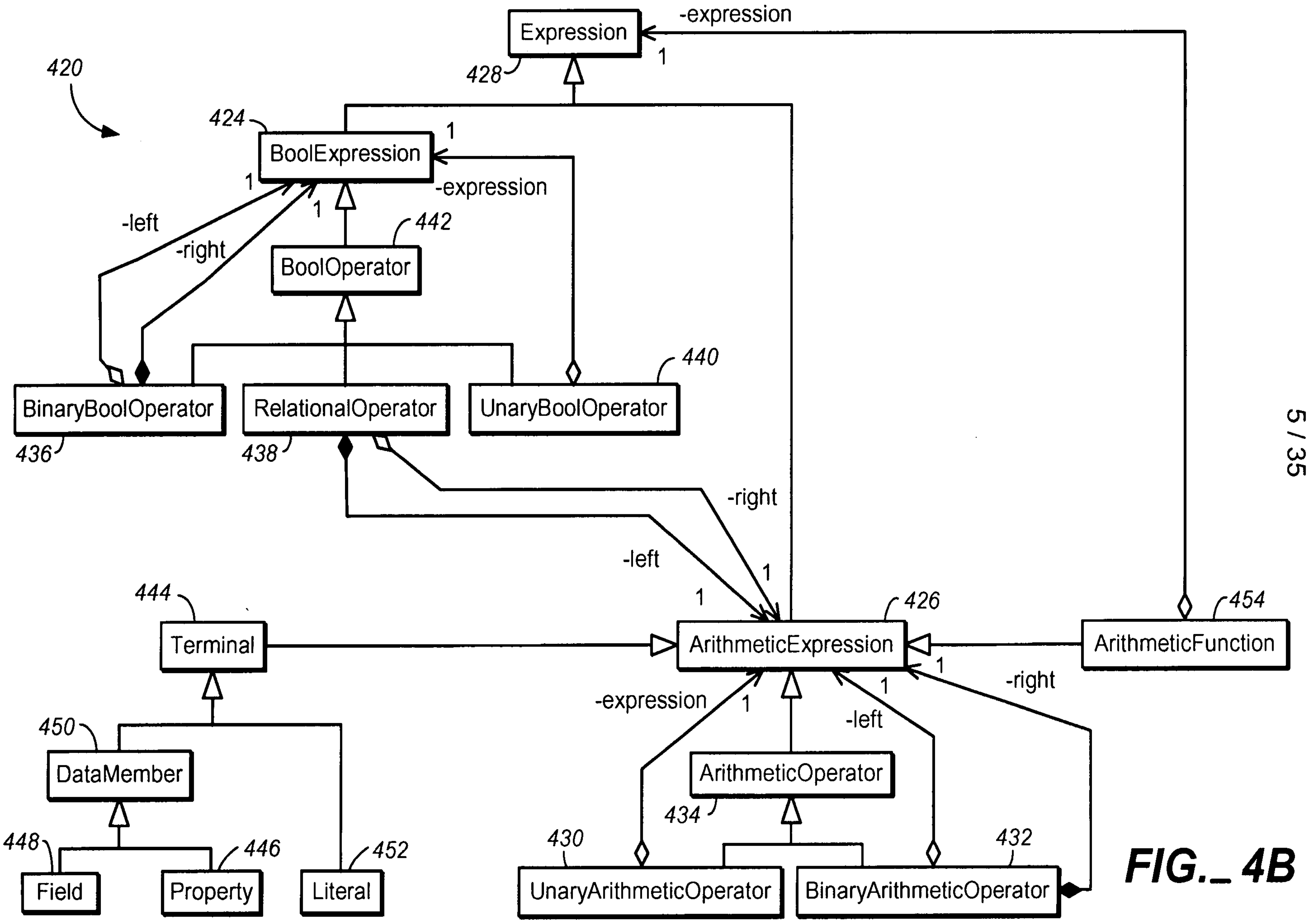

420
Expression
428
-expression
1
424
BoolExpression
1
-expression
1
1
-left
-right
442
BoolOperator
440
BinaryBoolOperator
RelationalOperator
UnaryBoolOperator
436
438
-right
-left
1
1
444
Terminal
426
ArithmeticExpression
454
ArithmeticFunction
450
DataMember
448
Field
Property
446
Literal
452
-expression
1
1
-left
1
-right
ArithmeticOperator
434
430
432
UnaryArithmeticOperator
BinaryArithmeticOperator

FIG._ 4B

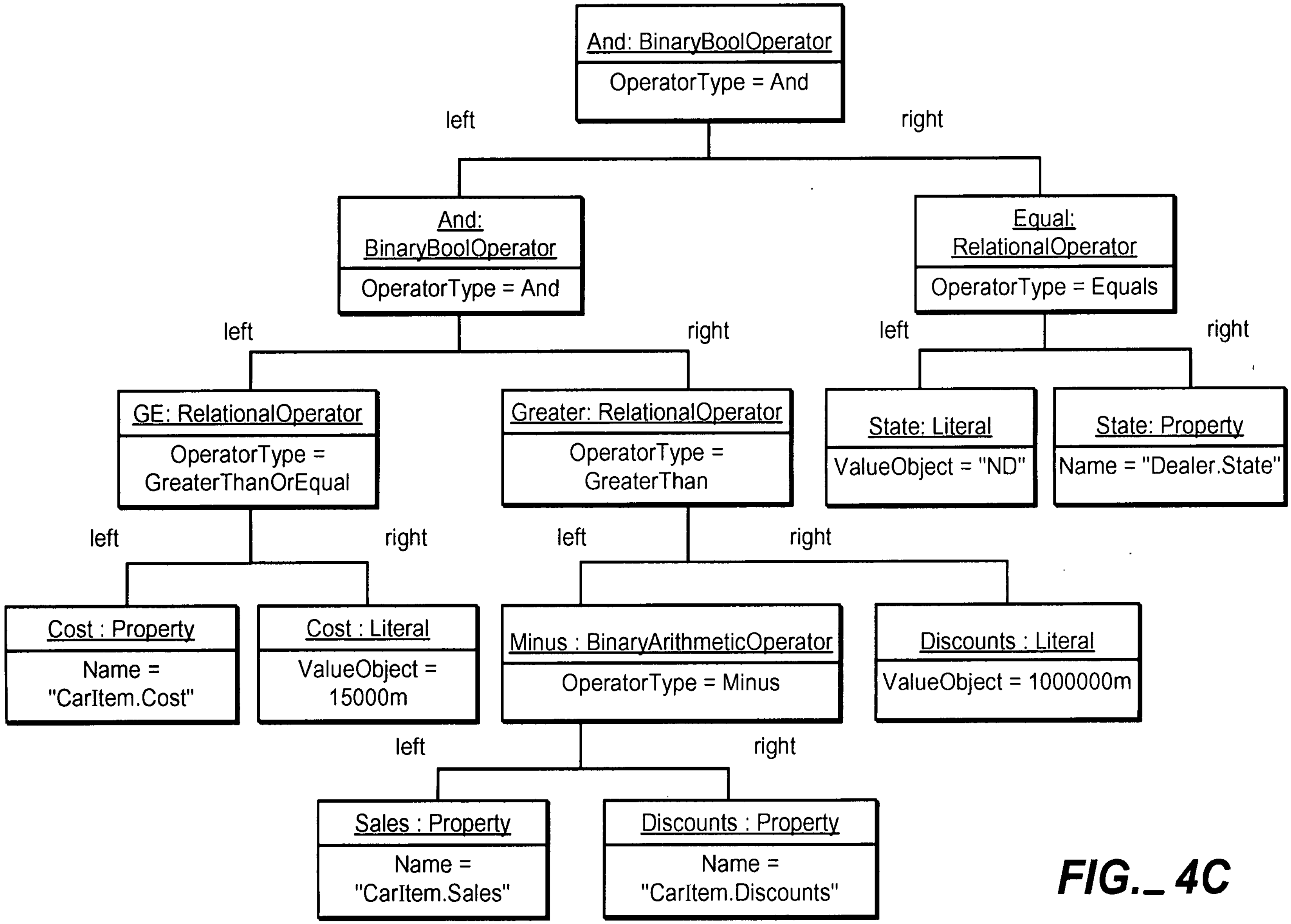

And: BinaryBoolOperator
OperatorType = And
left
right
And: BinaryBoolOperator
OperatorType = And
Equal: RelationalOperator
OperatorType = Equals
left
right
left
right
GE: RelationalOperator
OperatorType = GreaterThanOrEqual
Greater: RelationalOperator
OperatorType = GreaterThan
State: Literal
ValueObject = "ND"
State: Property
Name = "Dealer.State"
left
right
left
right
Cost : Property
Name = "CarItem.Cost"
Cost : Literal
ValueObject = 15000m
Minus : BinaryArithmeticOperator
OperatorType = Minus
Discounts : Literal
ValueObject = 1000000m
left
right
Sales : Property
Name = "CarItem.Sales"
Discounts : Property
Name = "CarItem.Discounts"

FIG._4C

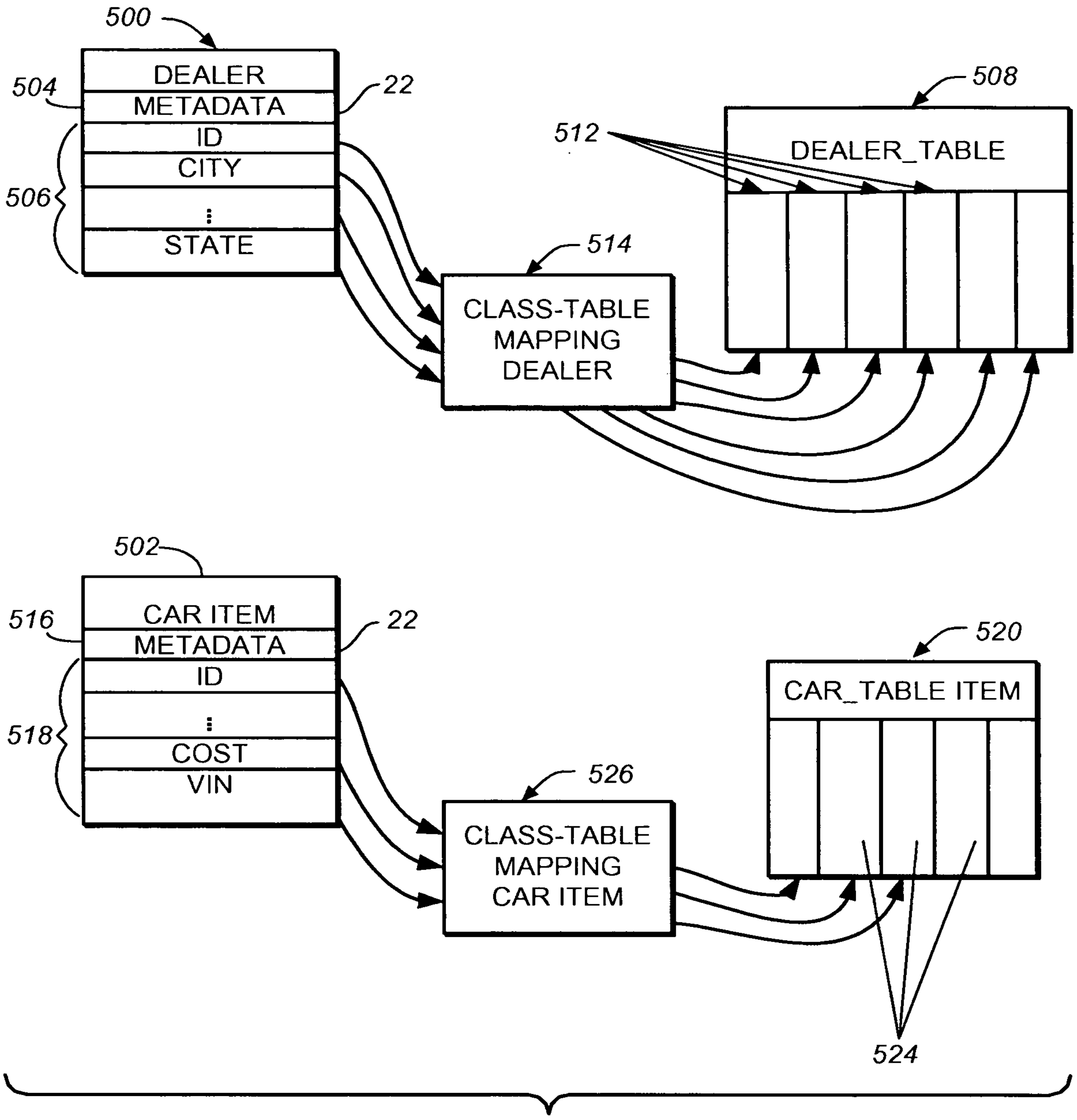

500
DEALER
504
METADATA
22
ID
CITY
506
⋮
STATE
508
512
DEALER_TABLE
514
CLASS-TABLE
MAPPING
DEALER
502
CAR ITEM
516
METADATA
22
ID
518
⋮
COST
VIN
520
CAR_TABLE ITEM
526
CLASS-TABLE
MAPPING
CAR ITEM
524

FIG._5

```
class CarItem {     // these properties are mapped to a database table
        public string ID
        public string Vin;
        public string Color;
        public decimal Cost;
        ... many others omitted ...
}
class Dealer {      // these properties are mapped to a different database table
        public string ID;
        public string City;
        public string State;
        ... many others omitted ...
}

AdHocQueryCriteria  adHocCriteria = Criteria.AdHocQueryCriteria(
        Criteria.EntityAliases(     // describes the objects involved in the query
                Criteria.EntityAlias(itemParentKey, typeof(CarItem))
                Criteria.EntityAlias(dealerParentKey, typeof(Dealer)) ),
        Criteria.JoinList(
                        /* entity to entity join */
                Criteria.InnerJoin("CarItem", "Dealer",
                (Property)"CarItem.DealerID" == (Property)"Dealer.ID")),
        Criteria.Select(            // the specific properties to retrieve
                (Property)"CarItem.ID",  // references the field in the above class
                (Property)"CarItem.Vin",
                (Property)"CarItem.Cost",
                (Property)"Dealer.ID",
                (Property)"Dealer.City",
                (Property)"Dealer.State"),
        Criteria.Where(
                (Property)"CarItem.Make" == "Geo" &&
                (Property)"CarItem.Model" == "Prism" &&
                (Property)"Dealer.State" == "ND")
        Criteria.OrderBy((Property)"Dealer.Cost"));
```

FIG._6

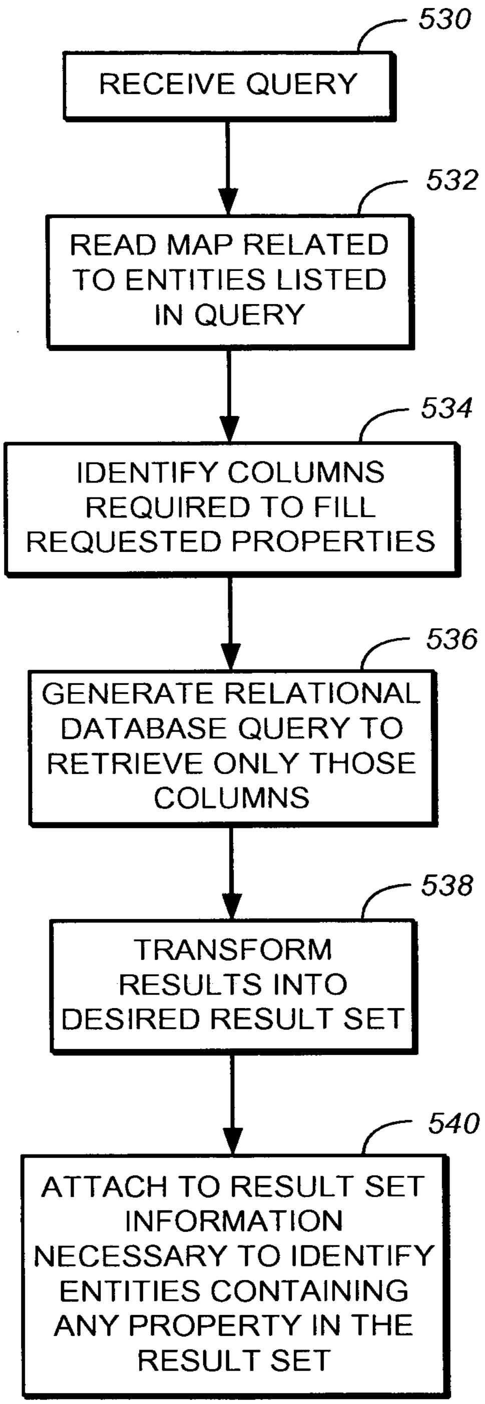

530
RECEIVE QUERY
532
READ MAP RELATED TO ENTITIES LISTED IN QUERY
534
IDENTIFY COLUMNS REQUIRED TO FILL REQUESTED PROPERTIES
536
GENERATE RELATIONAL DATABASE QUERY TO RETRIEVE ONLY THOSE COLUMNS
538
TRANSFORM RESULTS INTO DESIRED RESULT SET
540
ATTACH TO RESULT SET INFORMATION NECESSARY TO IDENTIFY ENTITIES CONTAINING ANY PROPERTY IN THE RESULT SET

FIG._7

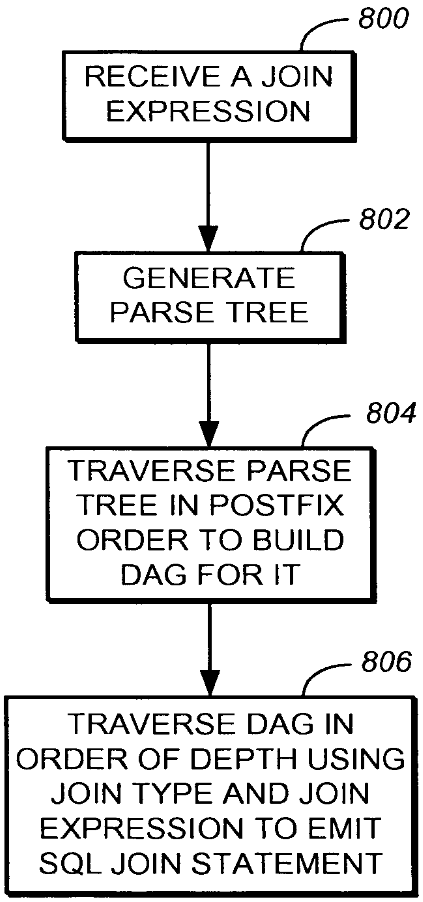

800
RECEIVE A JOIN EXPRESSION
802
GENERATE PARSE TREE
804
TRAVERSE PARSE TREE IN POSTFIX ORDER TO BUILD DAG FOR IT
806
TRAVERSE DAG IN ORDER OF DEPTH USING JOIN TYPE AND JOIN EXPRESSION TO EMIT SQL JOIN STATEMENT

FIG._8

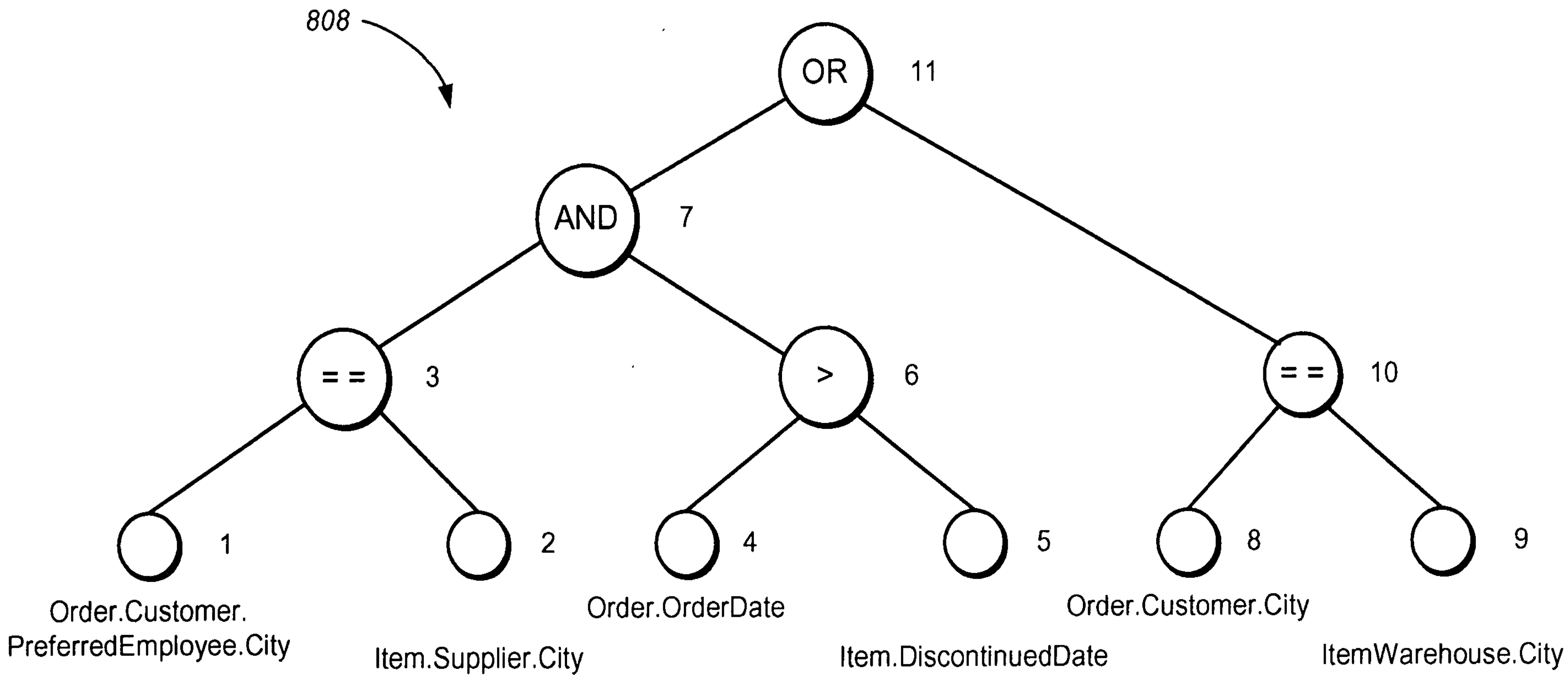
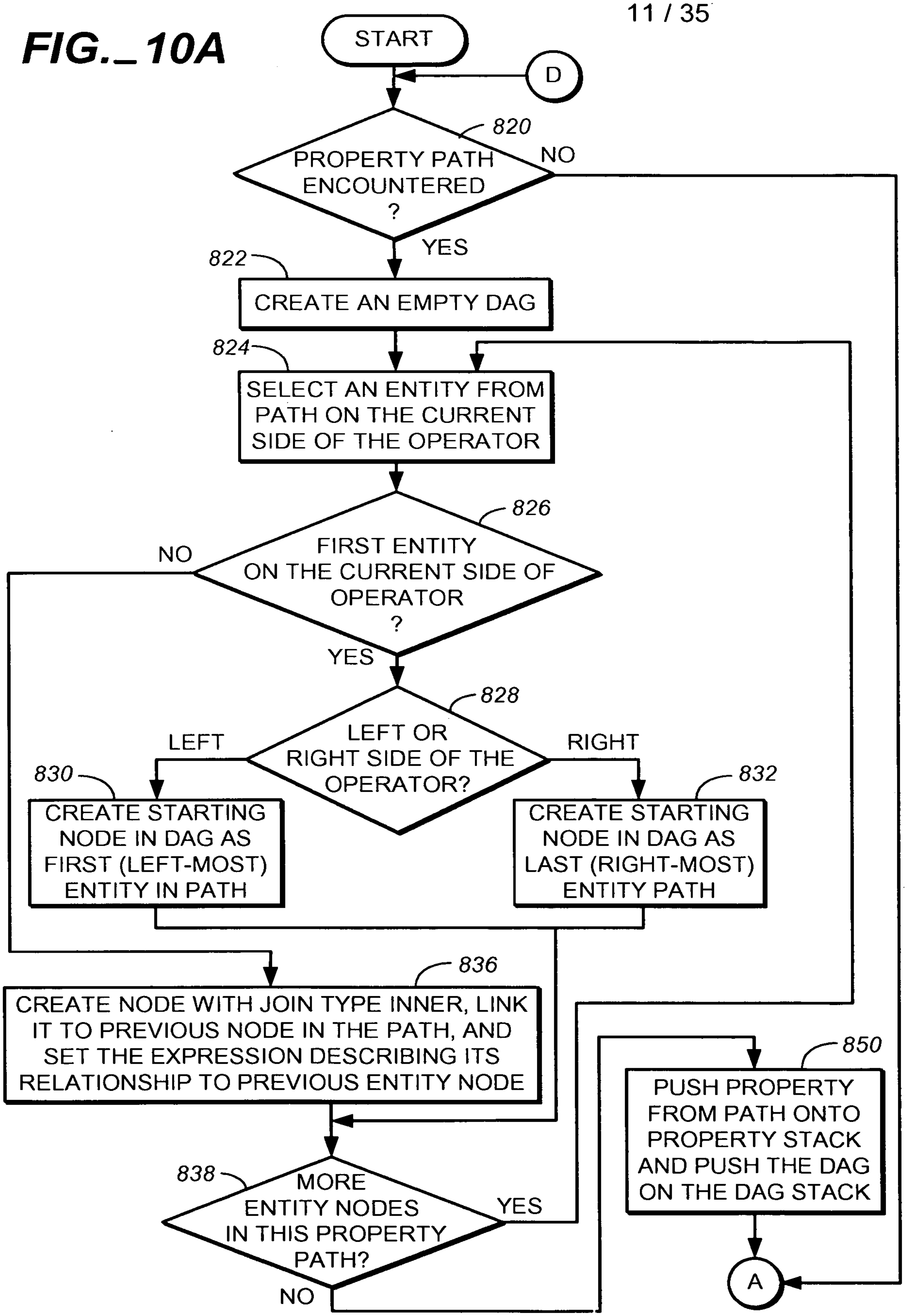

808
OR
11
AND
7
= =
3
>
6
= =
10
1
2
4
5
8
9
Order.Customer.
PreferredEmployee.City
Item.Supplier.City
Order.OrderDate
Item.DiscontinuedDate
Order.Customer.City
ItemWarehouse.City

FIG._ 9

FIG._10A


START
D
PROPERTY PATH ENCOUNTERED ?
820
NO
YES
822
CREATE AN EMPTY DAG
824
SELECT AN ENTITY FROM PATH ON THE CURRENT SIDE OF THE OPERATOR
826
FIRST ENTITY ON THE CURRENT SIDE OF OPERATOR ?
NO
YES
828
LEFT OR RIGHT SIDE OF THE OPERATOR?
LEFT
RIGHT
830
CREATE STARTING NODE IN DAG AS FIRST (LEFT-MOST) ENTITY IN PATH
832
CREATE STARTING NODE IN DAG AS LAST (RIGHT-MOST) ENTITY PATH
836
CREATE NODE WITH JOIN TYPE INNER, LINK IT TO PREVIOUS NODE IN THE PATH, AND SET THE EXPRESSION DESCRIBING ITS RELATIONSHIP TO PREVIOUS ENTITY NODE
850
PUSH PROPERTY FROM PATH ONTO PROPERTY STACK AND PUSH THE DAG ON THE DAG STACK
838
MORE ENTITY NODES IN THIS PROPERTY PATH?
YES
NO
A

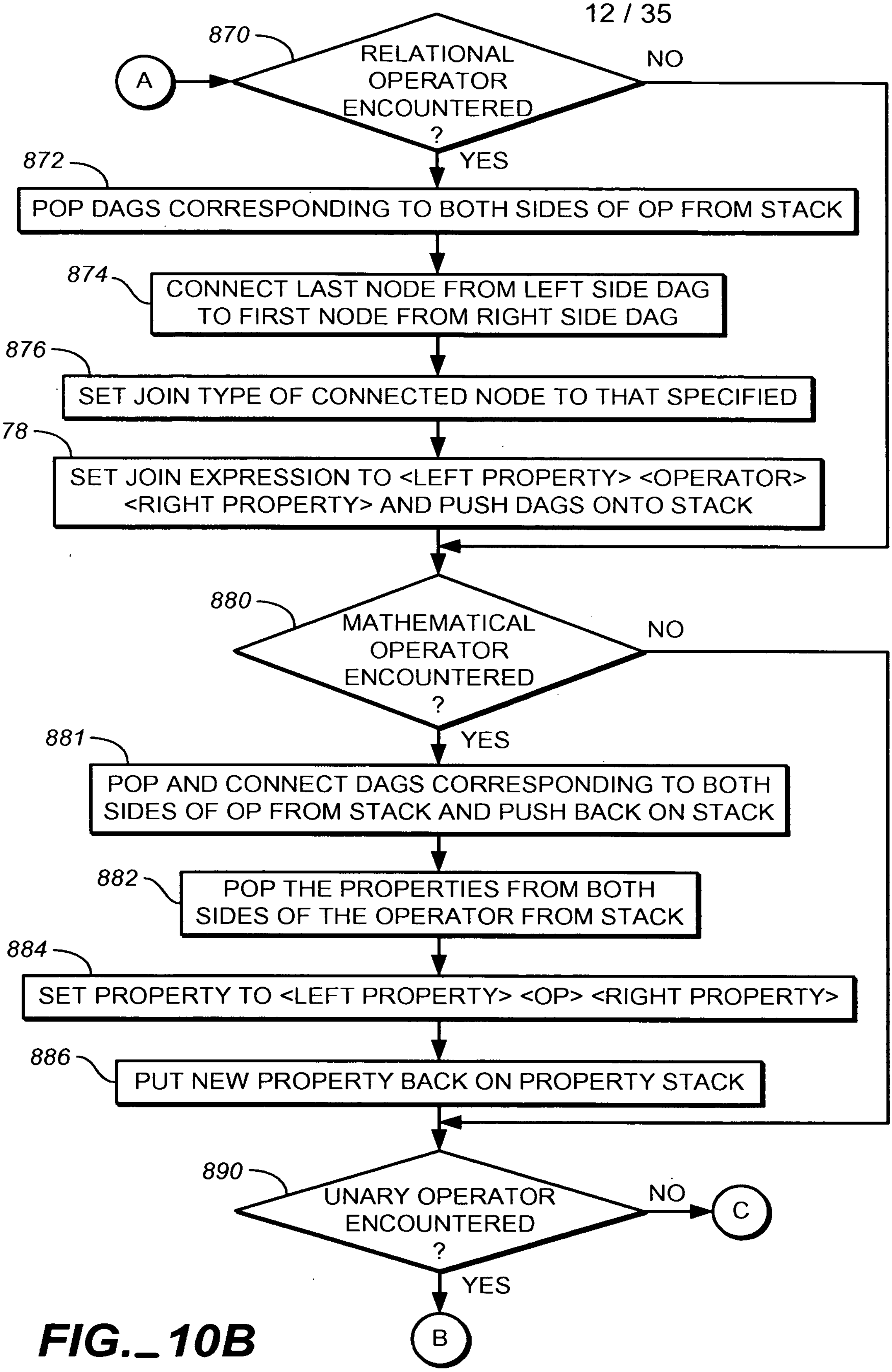

870
A
RELATIONAL OPERATOR ENCOUNTERED ?
NO
YES
872
POP DAGS CORRESPONDING TO BOTH SIDES OF OP FROM STACK
874
CONNECT LAST NODE FROM LEFT SIDE DAG TO FIRST NODE FROM RIGHT SIDE DAG
876
SET JOIN TYPE OF CONNECTED NODE TO THAT SPECIFIED
878
SET JOIN EXPRESSION TO <LEFT PROPERTY> <OPERATOR> <RIGHT PROPERTY> AND PUSH DAGS ONTO STACK
880
MATHEMATICAL OPERATOR ENCOUNTERED ?
NO
YES
881
POP AND CONNECT DAGS CORRESPONDING TO BOTH SIDES OF OP FROM STACK AND PUSH BACK ON STACK
882
POP THE PROPERTIES FROM BOTH SIDES OF THE OPERATOR FROM STACK
884
SET PROPERTY TO <LEFT PROPERTY> <OP> <RIGHT PROPERTY>
886
PUT NEW PROPERTY BACK ON PROPERTY STACK
890
UNARY OPERATOR ENCOUNTERED ?
NO
C
YES
B

FIG._10B

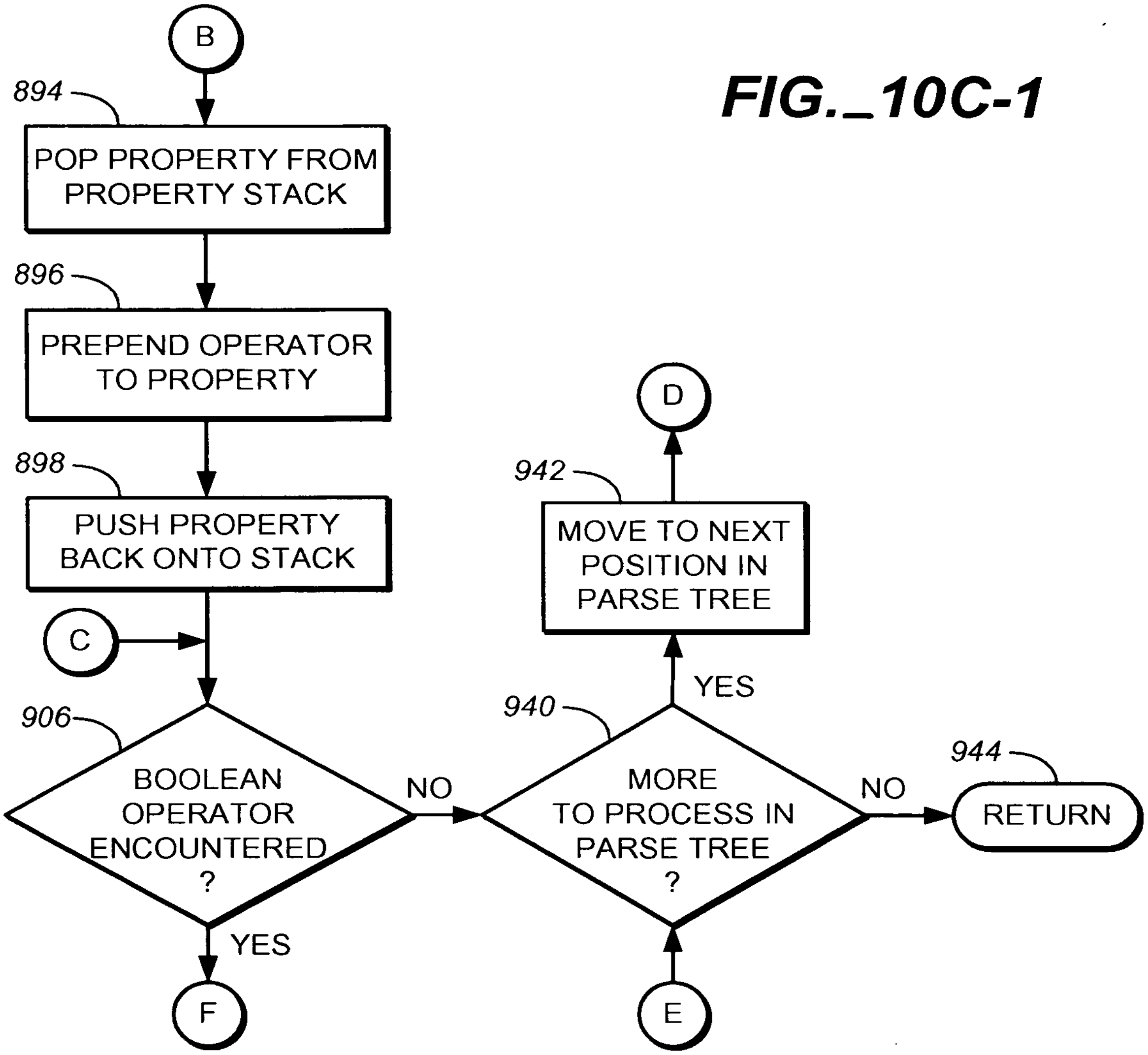

B
894
POP PROPERTY FROM PROPERTY STACK
896
PREPEND OPERATOR TO PROPERTY
898
PUSH PROPERTY BACK ONTO STACK
C
906
BOOLEAN OPERATOR ENCOUNTERED ?
YES
F
NO
940
MORE TO PROCESS IN PARSE TREE ?
YES
942
MOVE TO NEXT POSITION IN PARSE TREE
D
E
NO
944
RETURN

FIG._10C-1

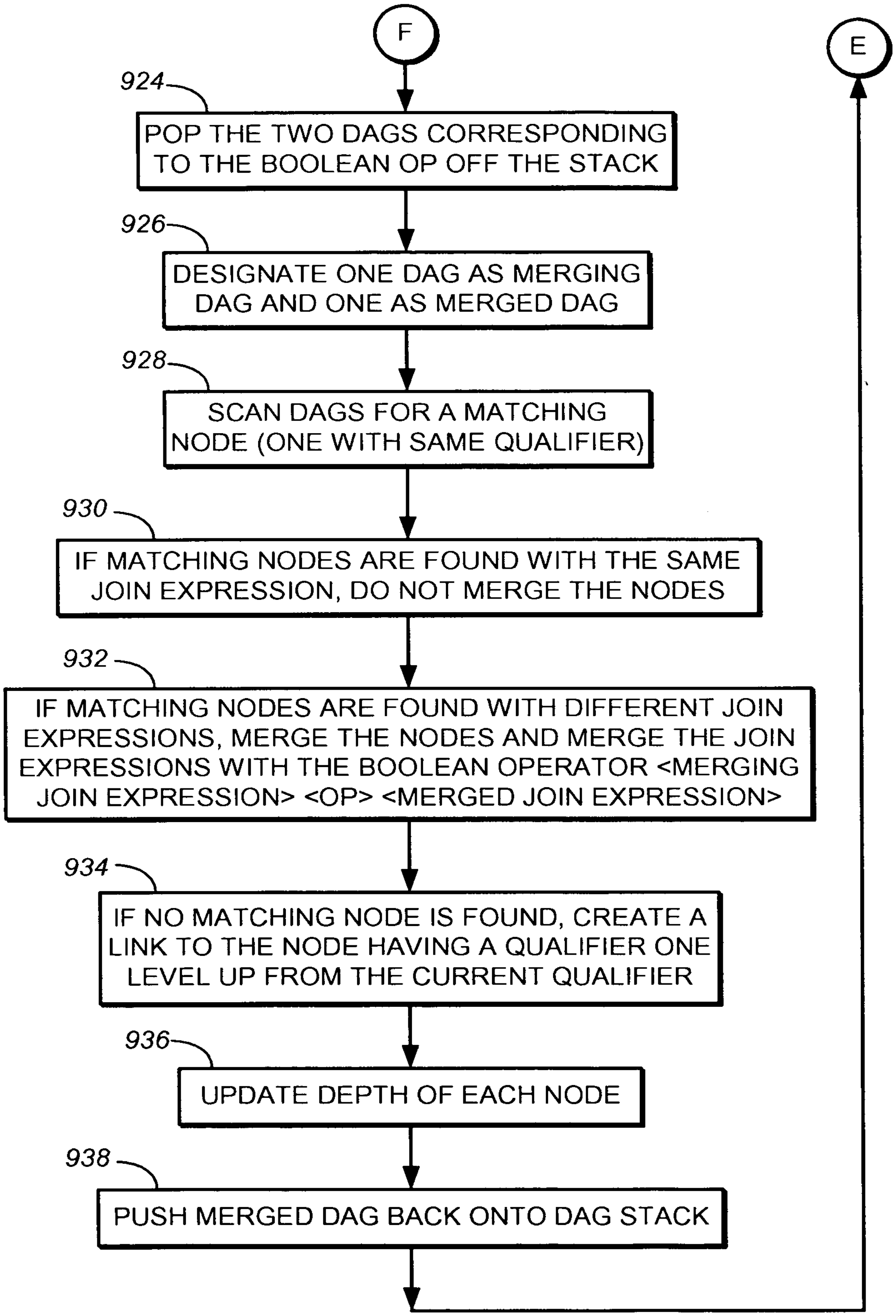

F
E
924
POP THE TWO DAGS CORRESPONDING TO THE BOOLEAN OP OFF THE STACK
926
DESIGNATE ONE DAG AS MERGING DAG AND ONE AS MERGED DAG
928
SCAN DAGS FOR A MATCHING NODE (ONE WITH SAME QUALIFIER)
930
IF MATCHING NODES ARE FOUND WITH THE SAME JOIN EXPRESSION, DO NOT MERGE THE NODES
932
IF MATCHING NODES ARE FOUND WITH DIFFERENT JOIN EXPRESSIONS, MERGE THE NODES AND MERGE THE JOIN EXPRESSIONS WITH THE BOOLEAN OPERATOR <MERGING JOIN EXPRESSION> <OP> <MERGED JOIN EXPRESSION>
934
IF NO MATCHING NODE IS FOUND, CREATE A LINK TO THE NODE HAVING A QUALIFIER ONE LEVEL UP FROM THE CURRENT QUALIFIER
936
UPDATE DEPTH OF EACH NODE
938
PUSH MERGED DAG BACK ONTO DAG STACK

FIG._10C-2

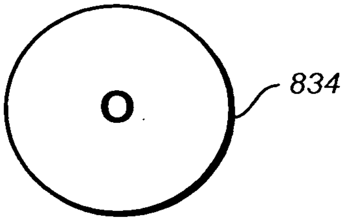


FIG._11A

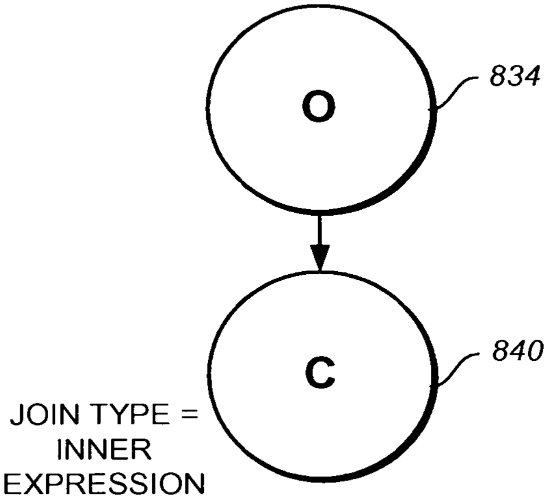


FIG._11B

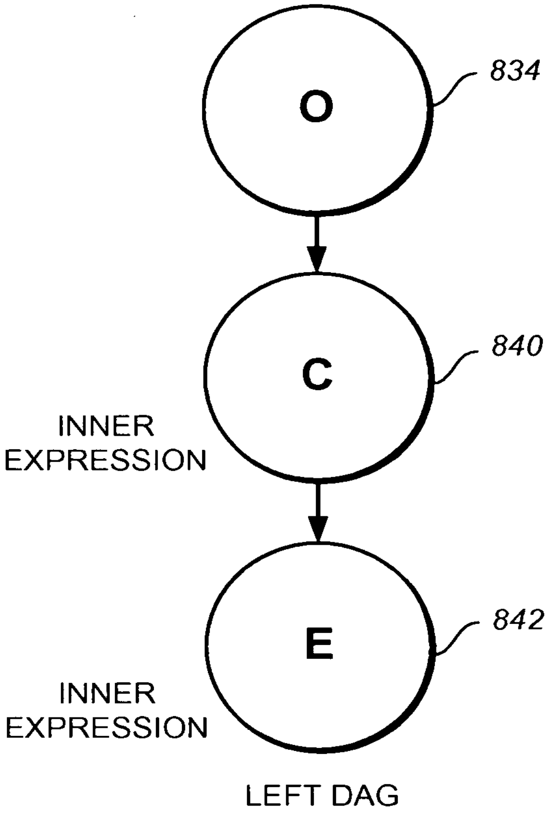


FIG._11C

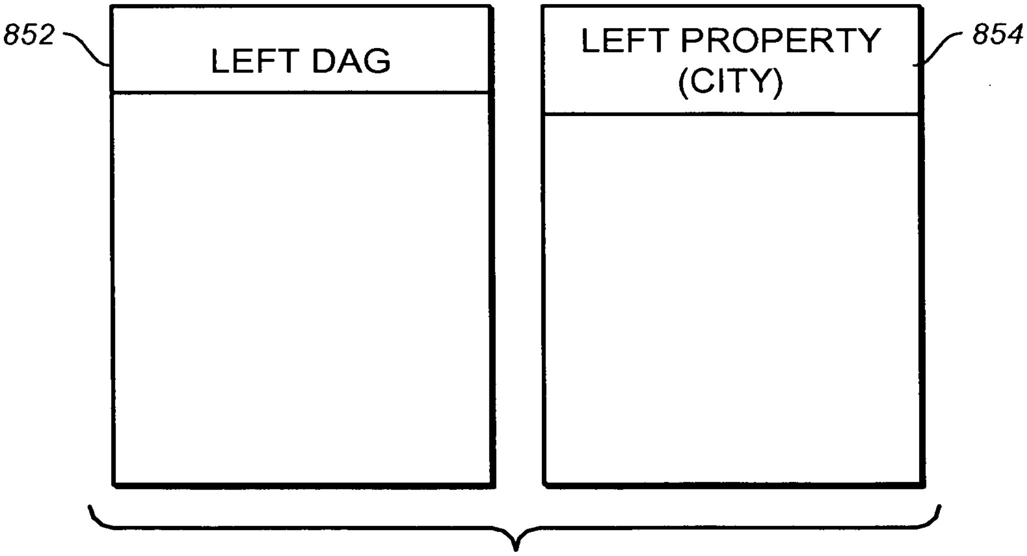


FIG._11D

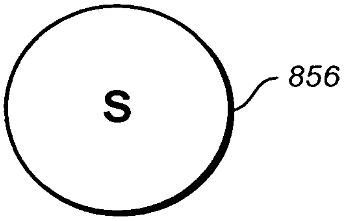


FIG._11E

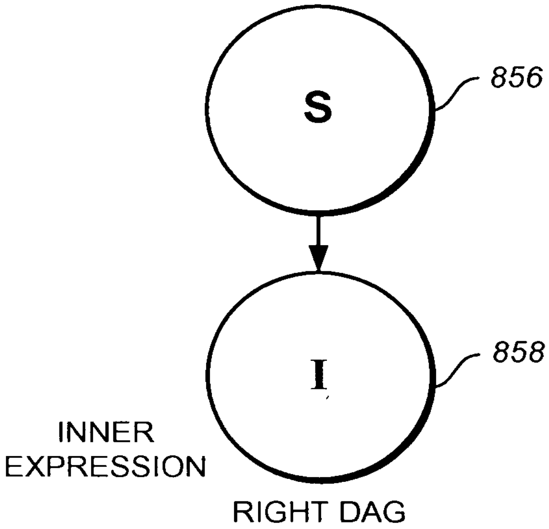


FIG._11F

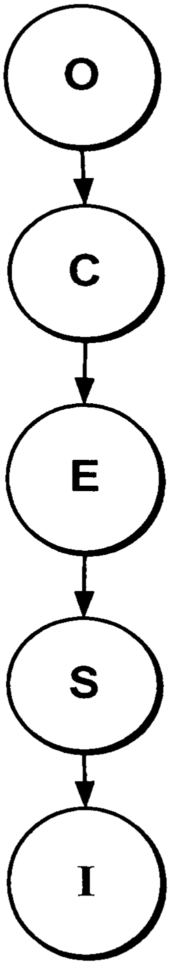


FIG._12

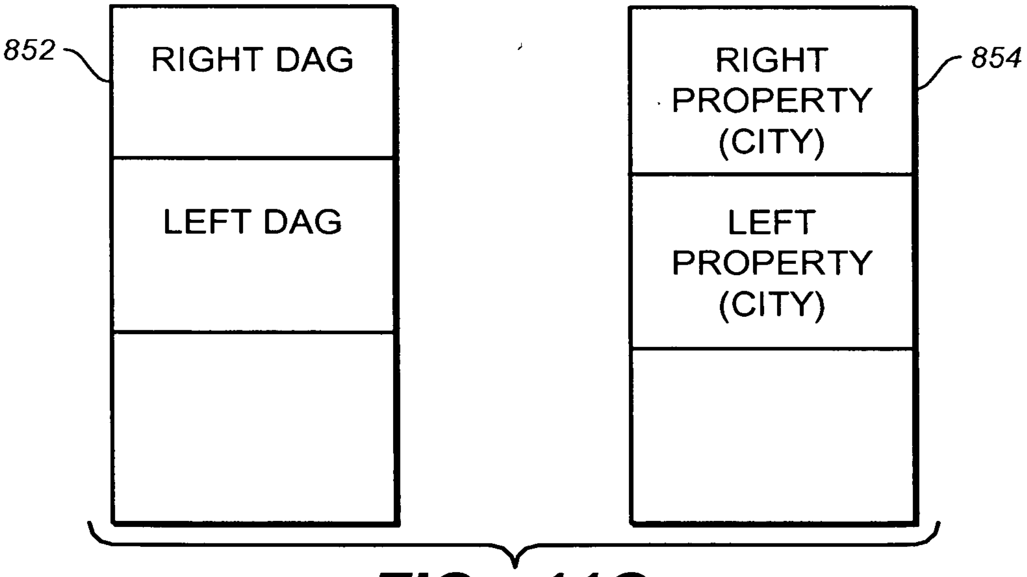


FIG._11G

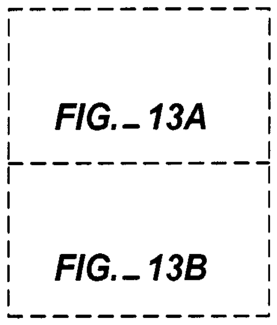


FIG._13

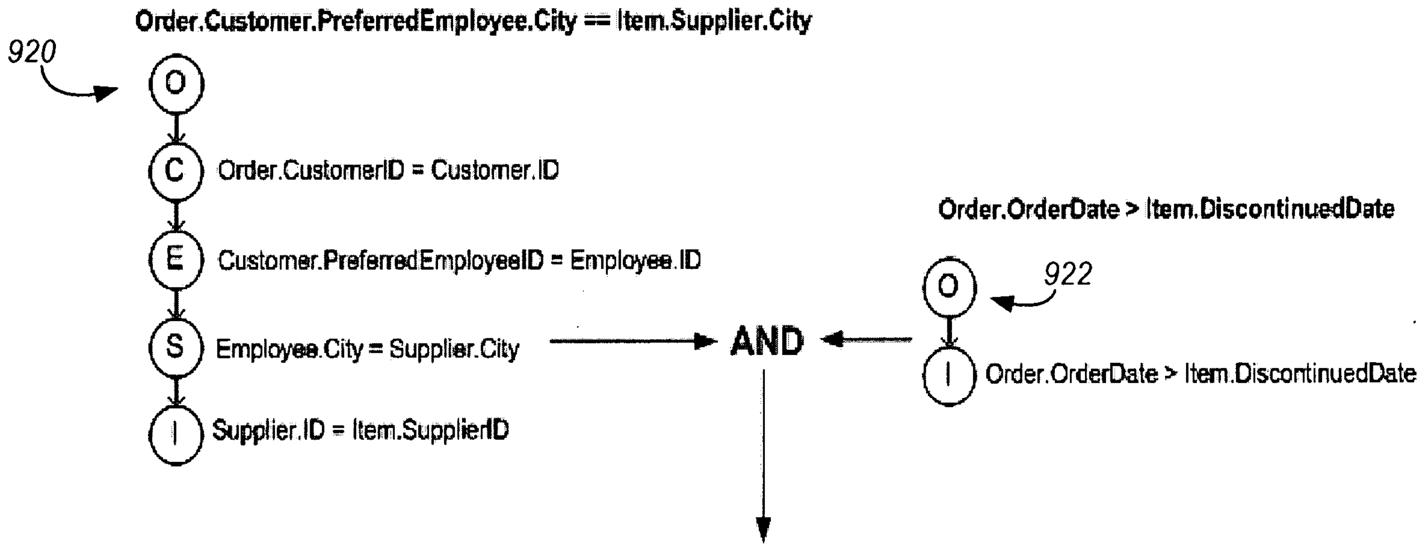

Order.Customer.PreferredEmployee.City == Item.Supplier.City
920
O
C
Order.CustomerID = Customer.ID
E
Customer.PreferredEmployeeID = Employee.ID
S
Employee.City = Supplier.City
I
Supplier.ID = Item.SupplierID
AND
Order.OrderDate > Item.DiscontinuedDate
O
922
I
Order.OrderDate > Item.DiscontinuedDate

FIG._13A

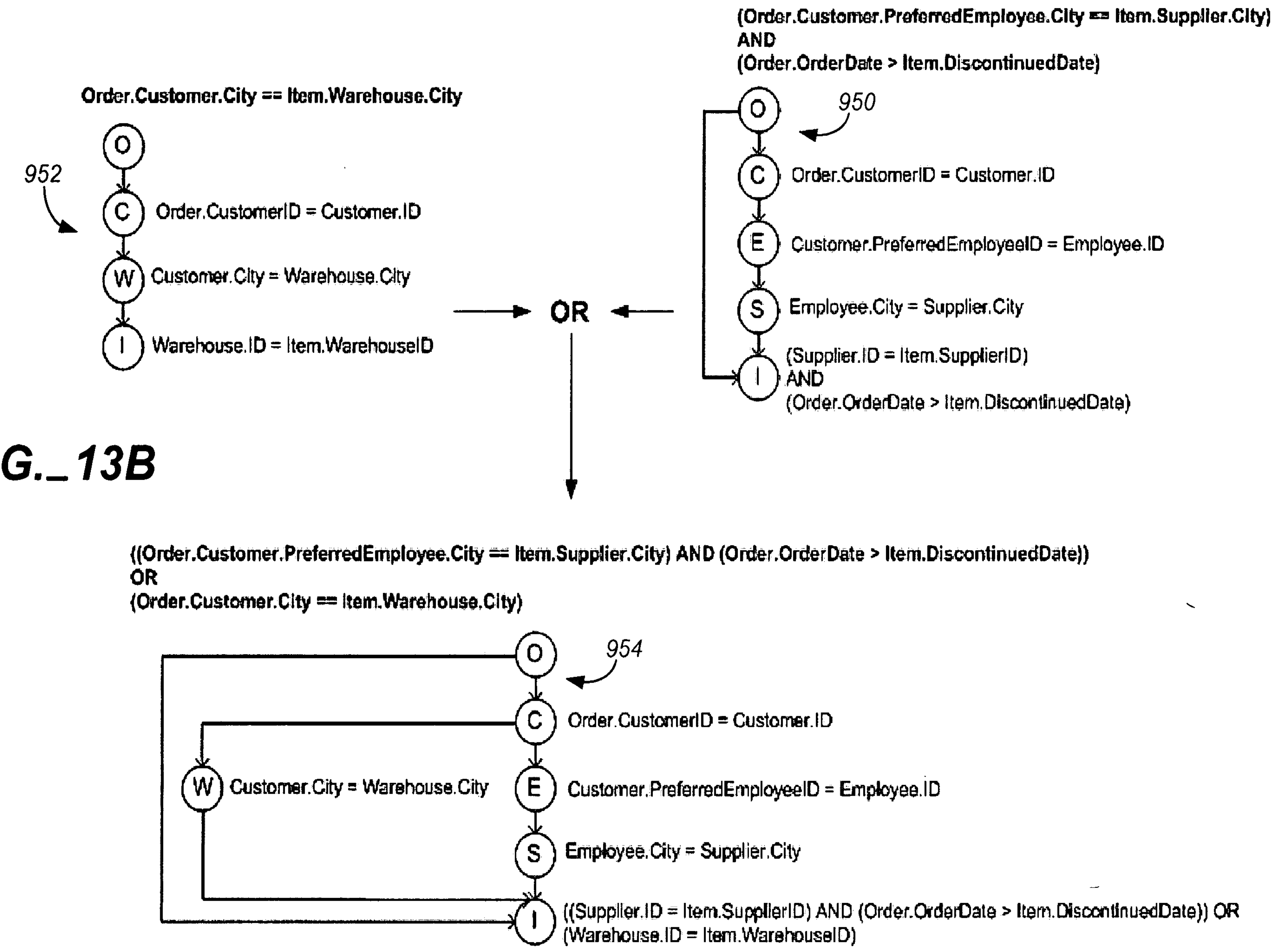

(Order.Customer.PreferredEmployee.City == Item.Supplier.City)
AND
(Order.OrderDate > Item.DiscontinuedDate)
Order.Customer.City == Item.Warehouse.City
950
952
O
C
Order.CustomerID = Customer.ID
W
Customer.City = Warehouse.City
I
Warehouse.ID = Item.WarehouseID
OR
O
C
Order.CustomerID = Customer.ID
E
Customer.PreferredEmployeeID = Employee.ID
S
Employee.City = Supplier.City
I
(Supplier.ID = Item.SupplierID)
AND
(Order.OrderDate > Item.DiscontinuedDate)
((Order.Customer.PreferredEmployee.City == Item.Supplier.City) AND (Order.OrderDate > Item.DiscontinuedDate))
OR
(Order.Customer.City == Item.Warehouse.City)
954
O
C
Order.CustomerID = Customer.ID
W
Customer.City = Warehouse.City
E
Customer.PreferredEmployeeID = Employee.ID
S
Employee.City = Supplier.City
I
((Supplier.ID = Item.SupplierID) AND (Order.OrderDate > Item.DiscontinuedDate)) OR
(Warehouse.ID = Item.WarehouseID)

FIG._13B

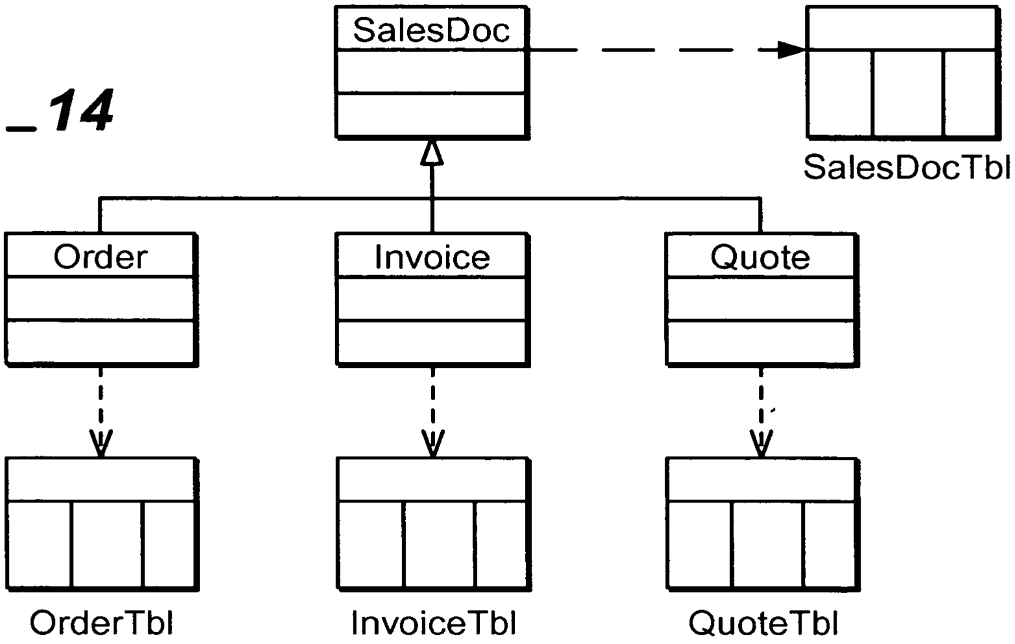


FIG._14

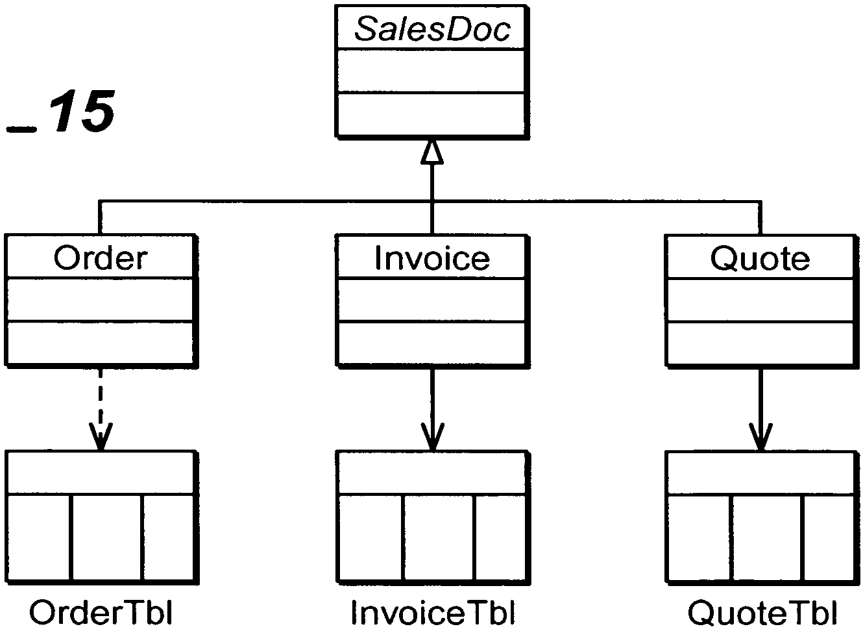


FIG._15

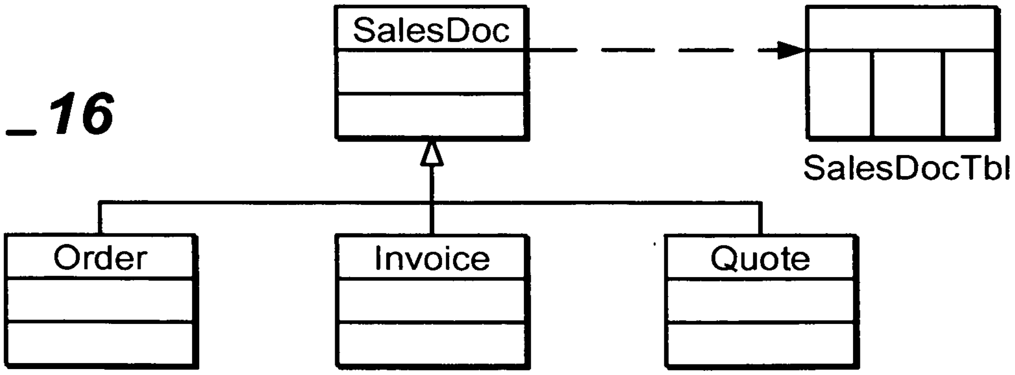


FIG._16

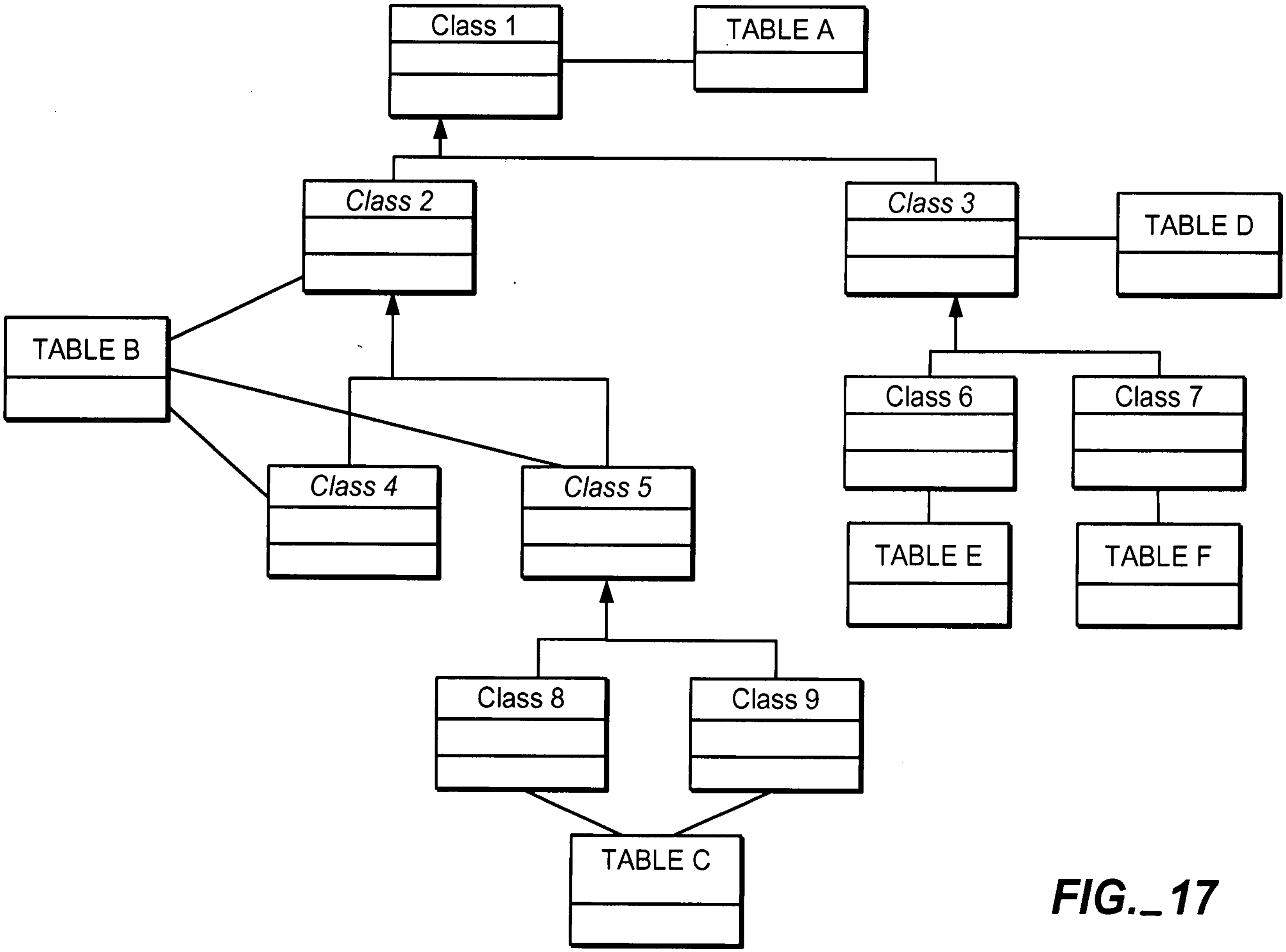

Class 1
TABLE A
Class 2
Class 3
TABLE D
TABLE B
Class 6
Class 7
Class 4
Class 5
TABLE E
TABLE F
Class 8
Class 9
TABLE C

FIG._17

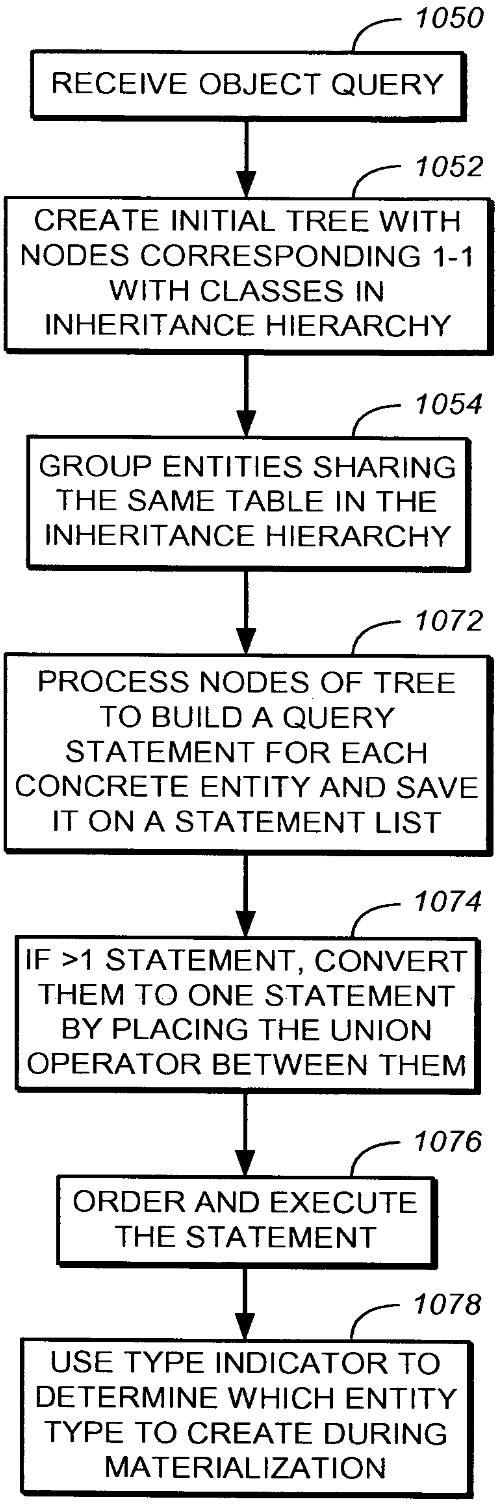

1050
RECEIVE OBJECT QUERY
1052
CREATE INITIAL TREE WITH NODES CORRESPONDING 1-1 WITH CLASSES IN INHERITANCE HIERARCHY
1054
GROUP ENTITIES SHARING THE SAME TABLE IN THE INHERITANCE HIERARCHY
1072
PROCESS NODES OF TREE TO BUILD A QUERY STATEMENT FOR EACH CONCRETE ENTITY AND SAVE IT ON A STATEMENT LIST
1074
IF >1 STATEMENT, CONVERT THEM TO ONE STATEMENT BY PLACING THE UNION OPERATOR BETWEEN THEM
1076
ORDER AND EXECUTE THE STATEMENT
1078
USE TYPE INDICATOR TO DETERMINE WHICH ENTITY TYPE TO CREATE DURING MATERIALIZATION

FIG._18

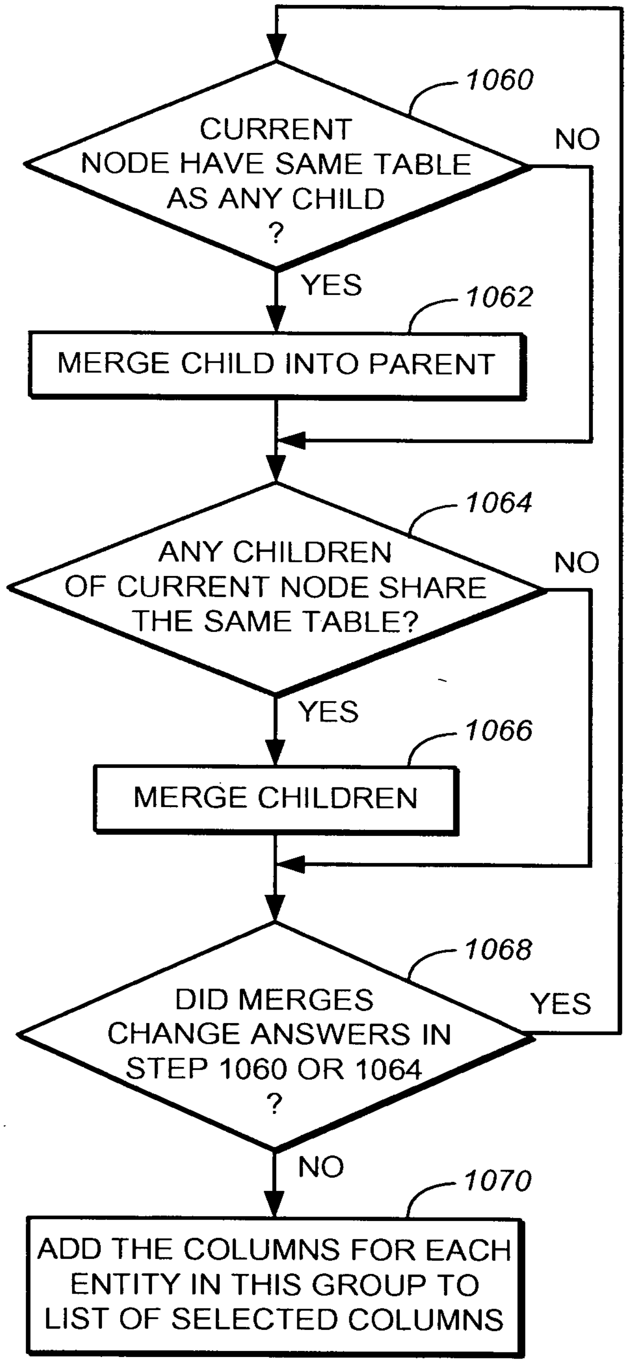

1060
CURRENT NODE HAVE SAME TABLE AS ANY CHILD ?
NO
YES
1062
MERGE CHILD INTO PARENT
1064
ANY CHILDREN OF CURRENT NODE SHARE THE SAME TABLE?
NO
YES
1066
MERGE CHILDREN
1068
DID MERGES CHANGE ANSWERS IN STEP 1060 OR 1064 ?
YES
NO
1070
ADD THE COLUMNS FOR EACH ENTITY IN THIS GROUP TO LIST OF SELECTED COLUMNS

FIG._18-1

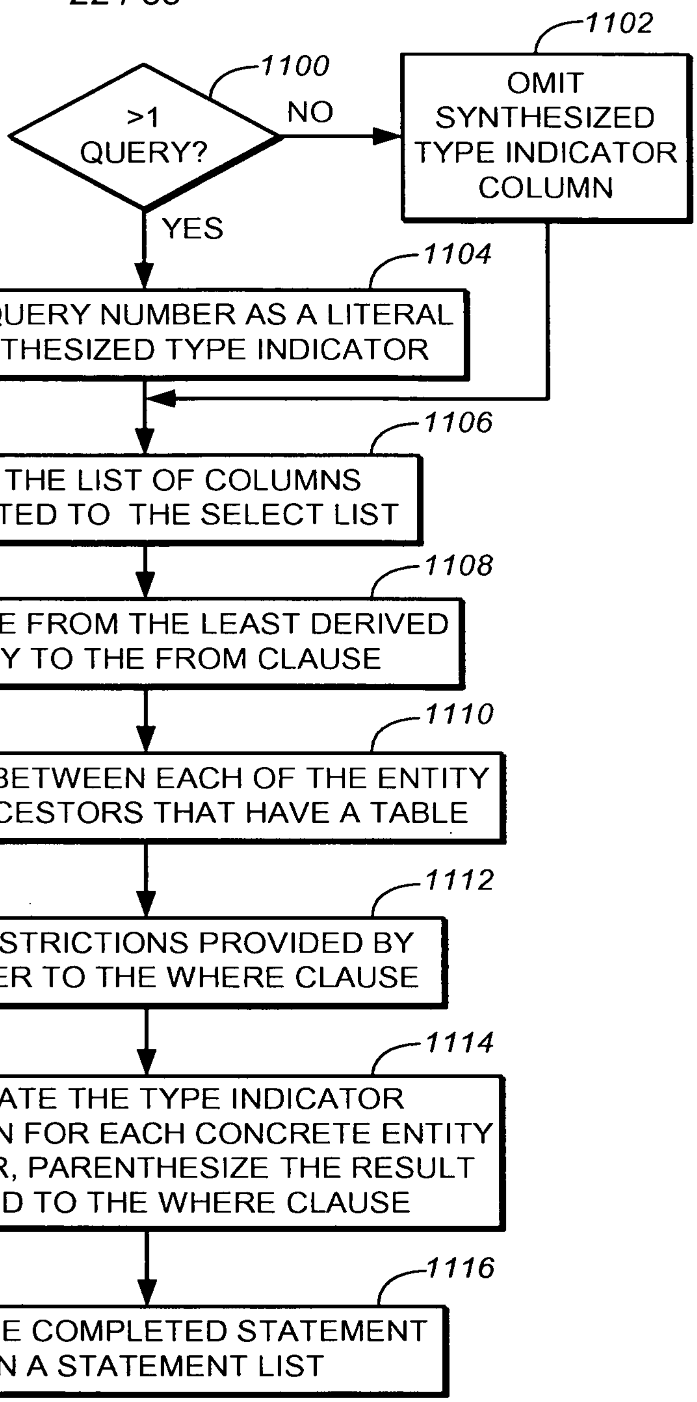

1100
>1
QUERY?
NO
YES
1102
OMIT
SYNTHESIZED
TYPE INDICATOR
COLUMN
1104
SPECIFY QUERY NUMBER AS A LITERAL
FOR SYNTHESIZED TYPE INDICATOR
1106
ADD THE LIST OF COLUMNS
SELECTED TO THE SELECT LIST
1108
ADD TABLE FROM THE LEAST DERIVED
ENTITY TO THE FROM CLAUSE
1110
ADD A JOIN BETWEEN EACH OF THE ENTITY
GROUP ANCESTORS THAT HAVE A TABLE
1112
ADD RESTRICTIONS PROVIDED BY
THE USER TO THE WHERE CLAUSE
1114
SEPARATE THE TYPE INDICATOR
RESTRICTION FOR EACH CONCRETE ENTITY
WITH AN OR, PARENTHESIZE THE RESULT
AND ADD TO THE WHERE CLAUSE
1116
SAVE THE COMPLETED STATEMENT
ON A STATEMENT LIST

FIG._18-2

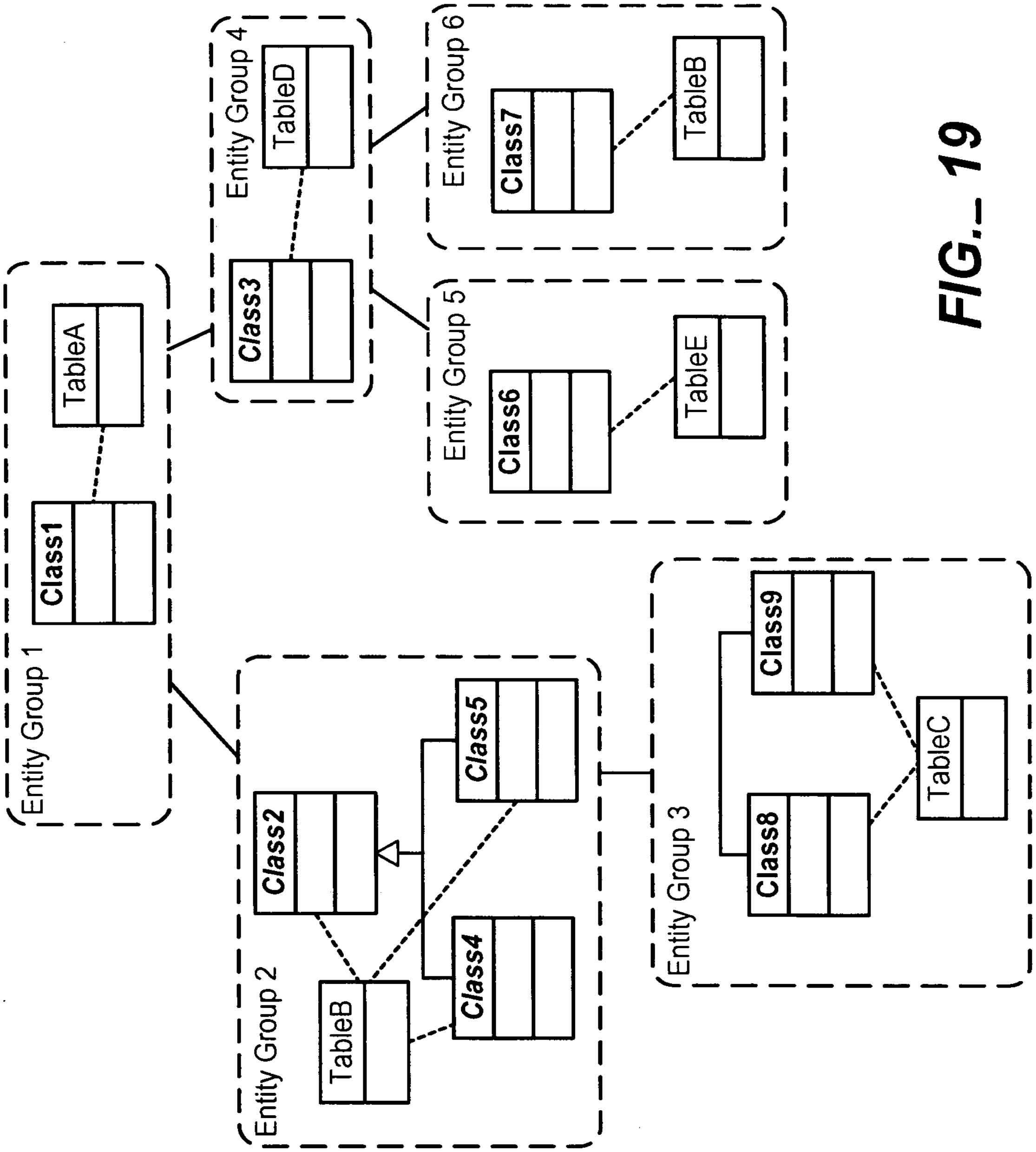

Entity Group 1
Class1
TableA
Entity Group 2
Class2
Class4
Class5
TableB
Entity Group 3
Class8
Class9
TableC
Entity Group 4
Class3
TableD
Entity Group 5
Class6
TableE
Entity Group 6
Class7
TableB

FIG._19

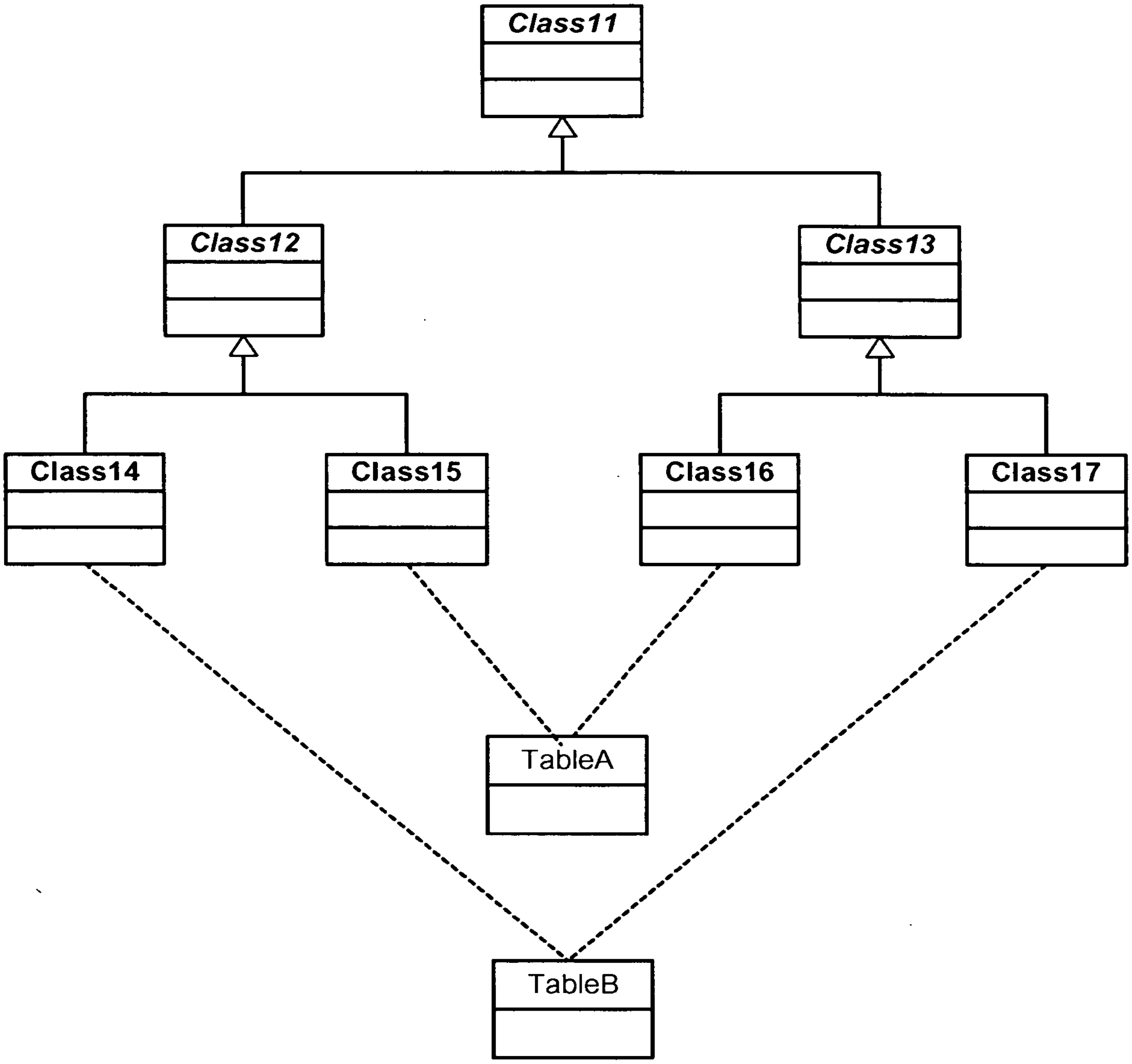

Class11
Class12
Class13
Class14
Class15
Class16
Class17
TableA
TableB

FIG._ 20

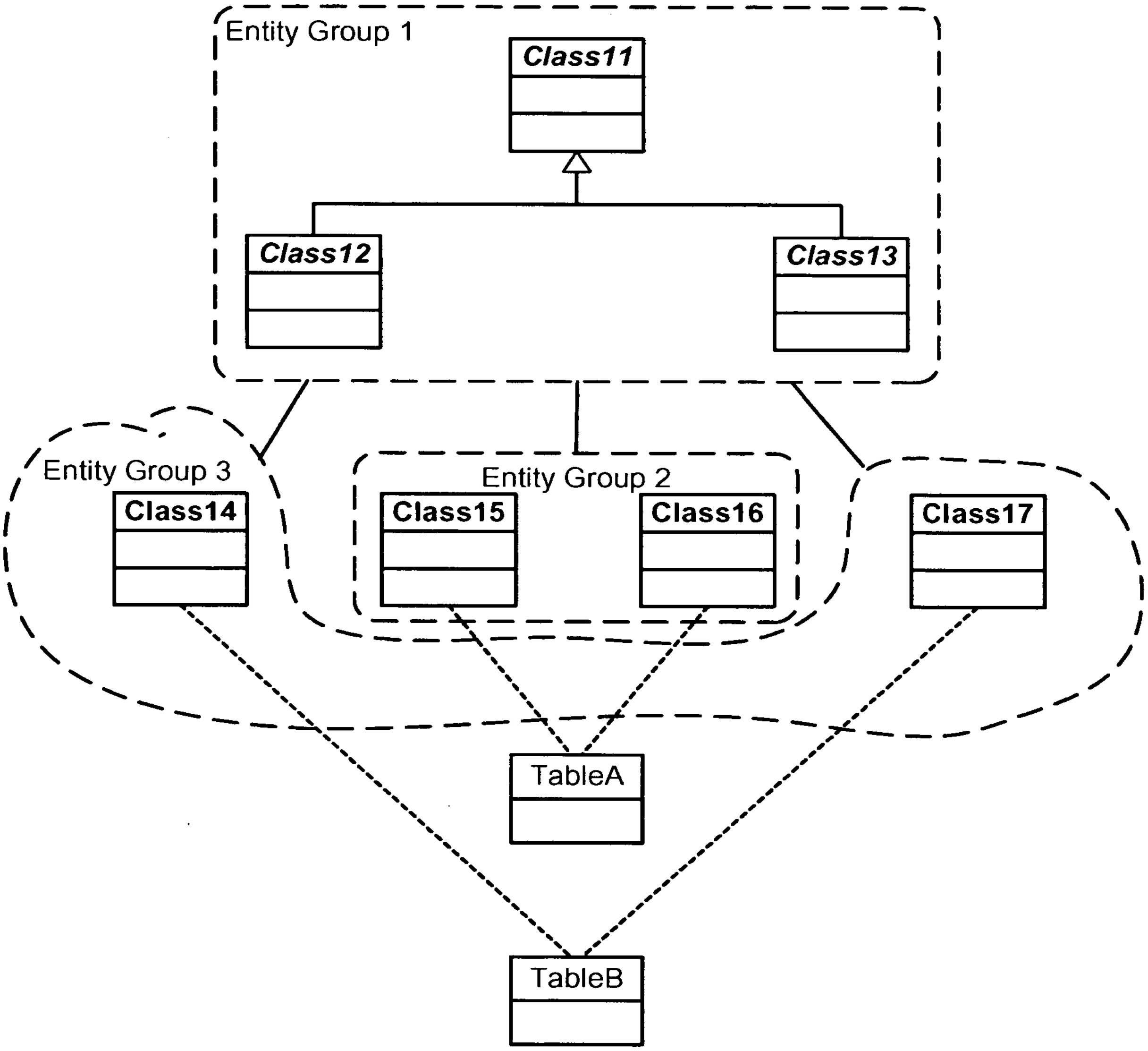

Entity Group 1
Class11
Class12
Class13
Entity Group 3
Class14
Entity Group 2
Class15
Class16
Class17
TableA
TableB

FIG._ 21

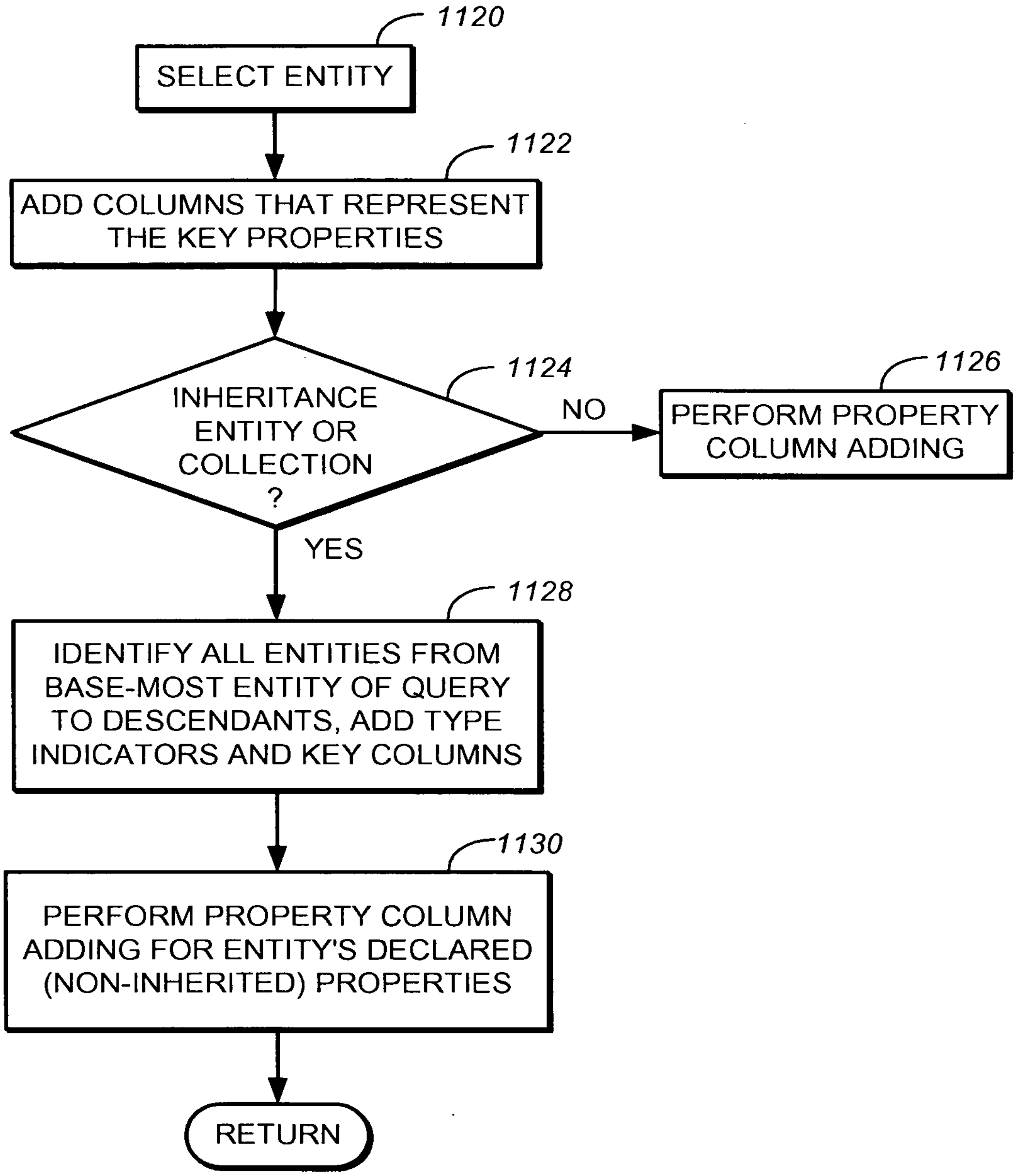

1120
SELECT ENTITY
1122
ADD COLUMNS THAT REPRESENT THE KEY PROPERTIES
1124
INHERITANCE ENTITY OR COLLECTION ?
NO
1126
PERFORM PROPERTY COLUMN ADDING
YES
1128
IDENTIFY ALL ENTITIES FROM BASE-MOST ENTITY OF QUERY TO DESCENDANTS, ADD TYPE INDICATORS AND KEY COLUMNS
1130
PERFORM PROPERTY COLUMN ADDING FOR ENTITY'S DECLARED (NON-INHERITED) PROPERTIES
RETURN

FIG._ 22

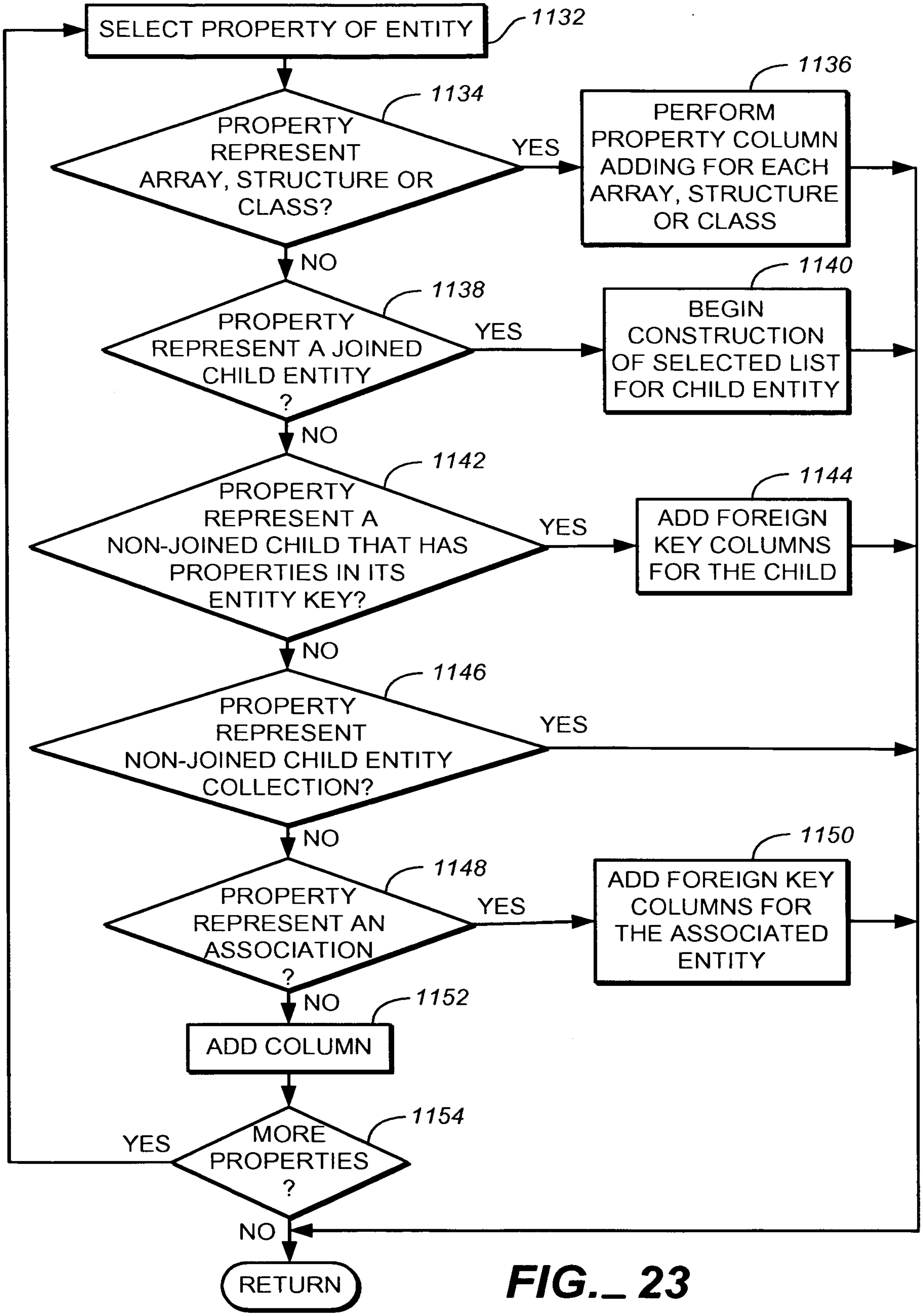

SELECT PROPERTY OF ENTITY
1132
PROPERTY REPRESENT ARRAY, STRUCTURE OR CLASS?
1134
YES
PERFORM PROPERTY COLUMN ADDING FOR EACH ARRAY, STRUCTURE OR CLASS
1136
NO
PROPERTY REPRESENT A JOINED CHILD ENTITY ?
1138
YES
BEGIN CONSTRUCTION OF SELECTED LIST FOR CHILD ENTITY
1140
NO
PROPERTY REPRESENT A NON-JOINED CHILD THAT HAS PROPERTIES IN ITS ENTITY KEY?
1142
YES
ADD FOREIGN KEY COLUMNS FOR THE CHILD
1144
NO
PROPERTY REPRESENT NON-JOINED CHILD ENTITY COLLECTION?
1146
YES
NO
PROPERTY REPRESENT AN ASSOCIATION ?
1148
YES
ADD FOREIGN KEY COLUMNS FOR THE ASSOCIATED ENTITY
1150
NO
ADD COLUMN
1152
MORE PROPERTIES ?
1154
YES
NO
RETURN

FIG._ 23

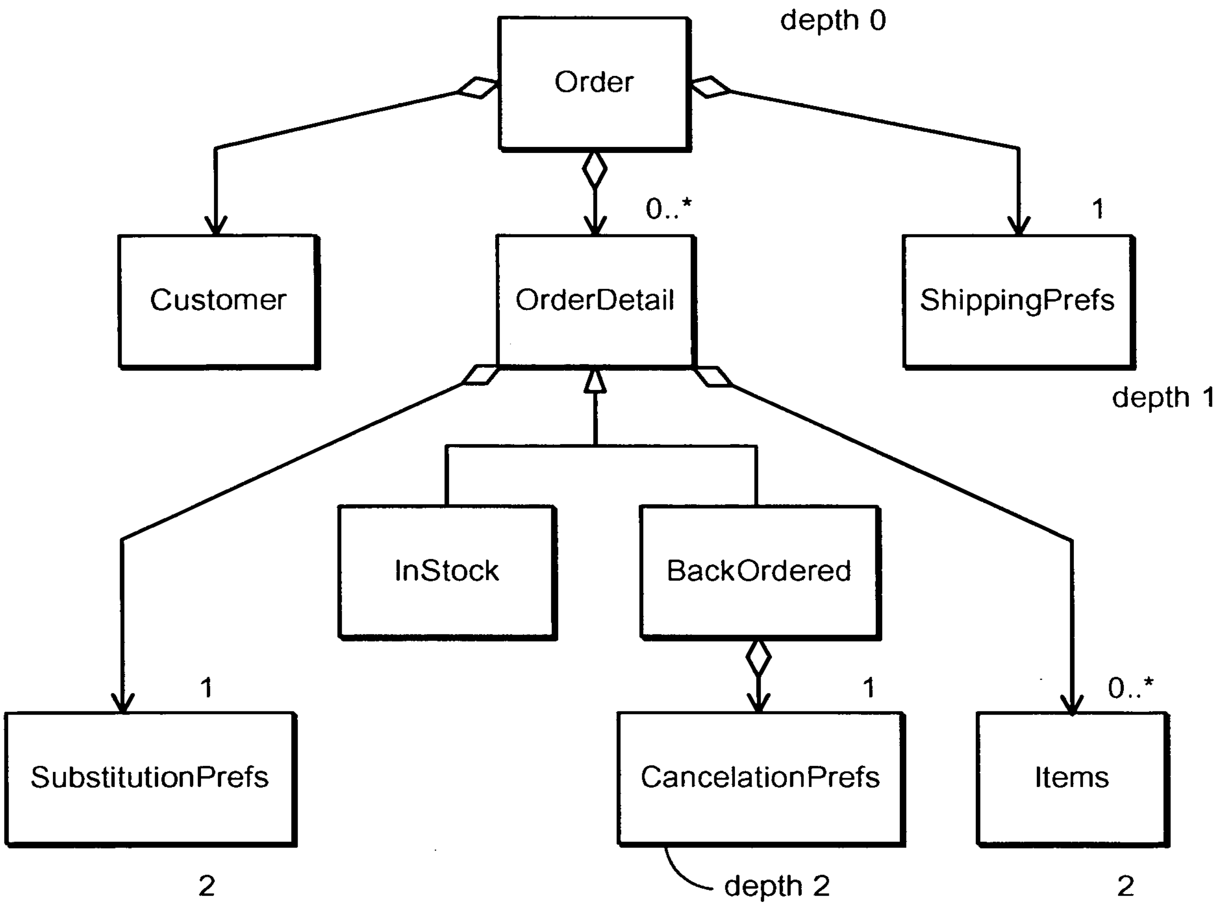

depth 0
Order
0..*
1
Customer
OrderDetail
ShippingPrefs
depth 1
InStock
BackOrdered
1
1
0..*
SubstitutionPrefs
CancelationPrefs
Items
2
depth 2
2

FIG._ 24

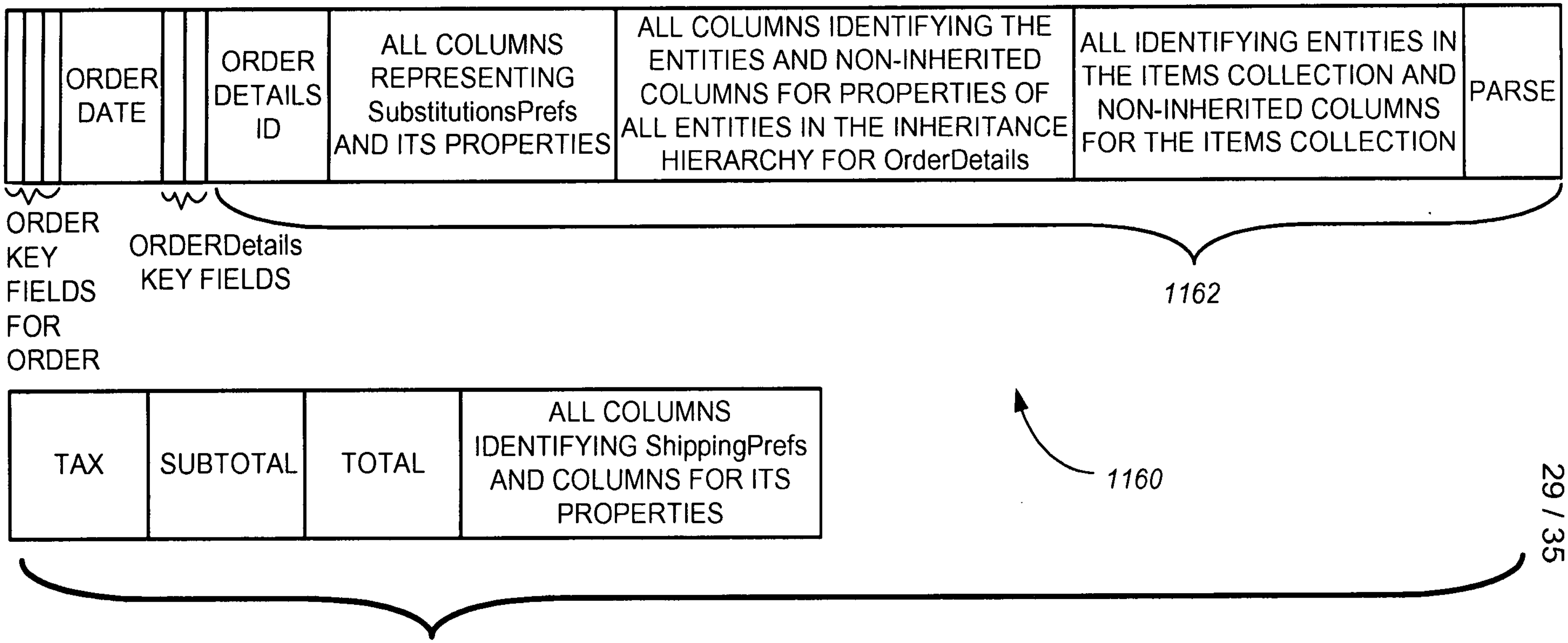

ORDER DATE
ORDER DETAILS ID
ALL COLUMNS REPRESENTING SubstitutionsPrefs AND ITS PROPERTIES
ALL COLUMNS IDENTIFYING THE ENTITIES AND NON-INHERITED COLUMNS FOR PROPERTIES OF ALL ENTITIES IN THE INHERITANCE HIERARCHY FOR OrderDetails
ALL IDENTIFYING ENTITIES IN THE ITEMS COLLECTION AND NON-INHERITED COLUMNS FOR THE ITEMS COLLECTION
PARSE
ORDER KEY FIELDS FOR ORDER
ORDERDetails KEY FIELDS
1162
1160
TAX
SUBTOTAL
TOTAL
ALL COLUMNS IDENTIFYING ShippingPrefs AND COLUMNS FOR ITS PROPERTIES

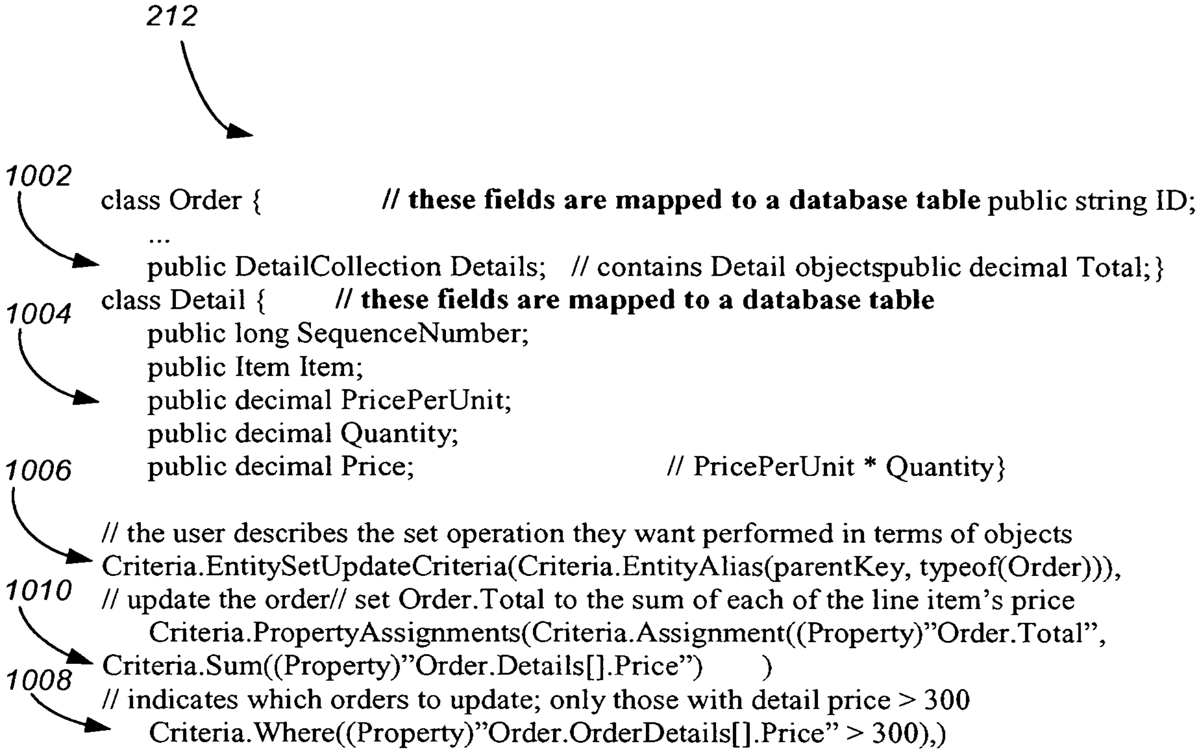
FIG._ 25

```
CLASS ORDER {
          ID
          DATE
          OrderDDetails
          Collection
          Tax
          Subtotal
          Total
          ShippingPrefs
}
```

FIG._25A

```
CLASS OrderDetail {
          ID
          SubstitutionPrefs
          Items
          Collection
          Misc
}
```

FIG._25B

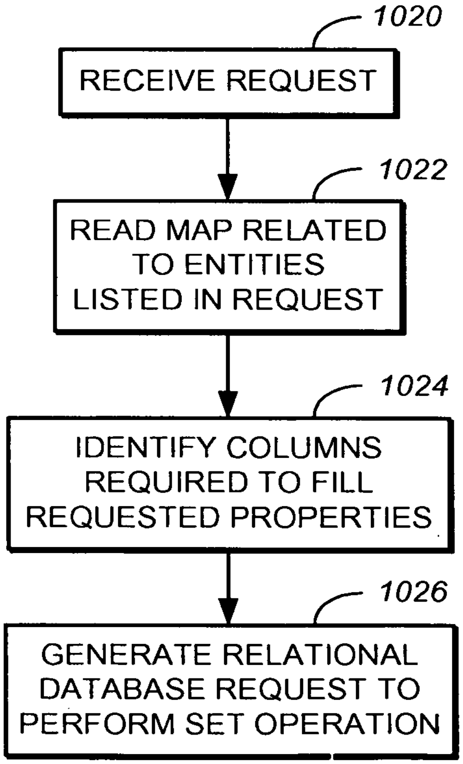


212
1002
1004
1006
1010
1008

```
class Order {           // these fields are mapped to a database table public string ID;
    ...
    public DetailCollection Details;  // contains Detail objectspublic decimal Total;}
class Detail {       // these fields are mapped to a database table
    public long SequenceNumber;
    public Item Item;
    public decimal PricePerUnit;
    public decimal Quantity;
    public decimal Price;                        // PricePerUnit * Quantity}

// the user describes the set operation they want performed in terms of objects
Criteria.EntitySetUpdateCriteria(Criteria.EntityAlias(parentKey, typeof(Order))),
// update the order// set Order.Total to the sum of each of the line item's price
    Criteria.PropertyAssignments(Criteria.Assignment((Property)"Order.Total",
Criteria.Sum((Property)"Order.Details[].Price")      )
// indicates which orders to update; only those with detail price > 300
    Criteria.Where((Property)"Order.OrderDetails[].Price" > 300),)
```

FIG._ 26



1020
RECEIVE REQUEST
1022
READ MAP RELATED TO ENTITIES LISTED IN REQUEST
1024
IDENTIFY COLUMNS REQUIRED TO FILL REQUESTED PROPERTIES
1026
GENERATE RELATIONAL DATABASE REQUEST TO PERFORM SET OPERATION

FIG._ 27

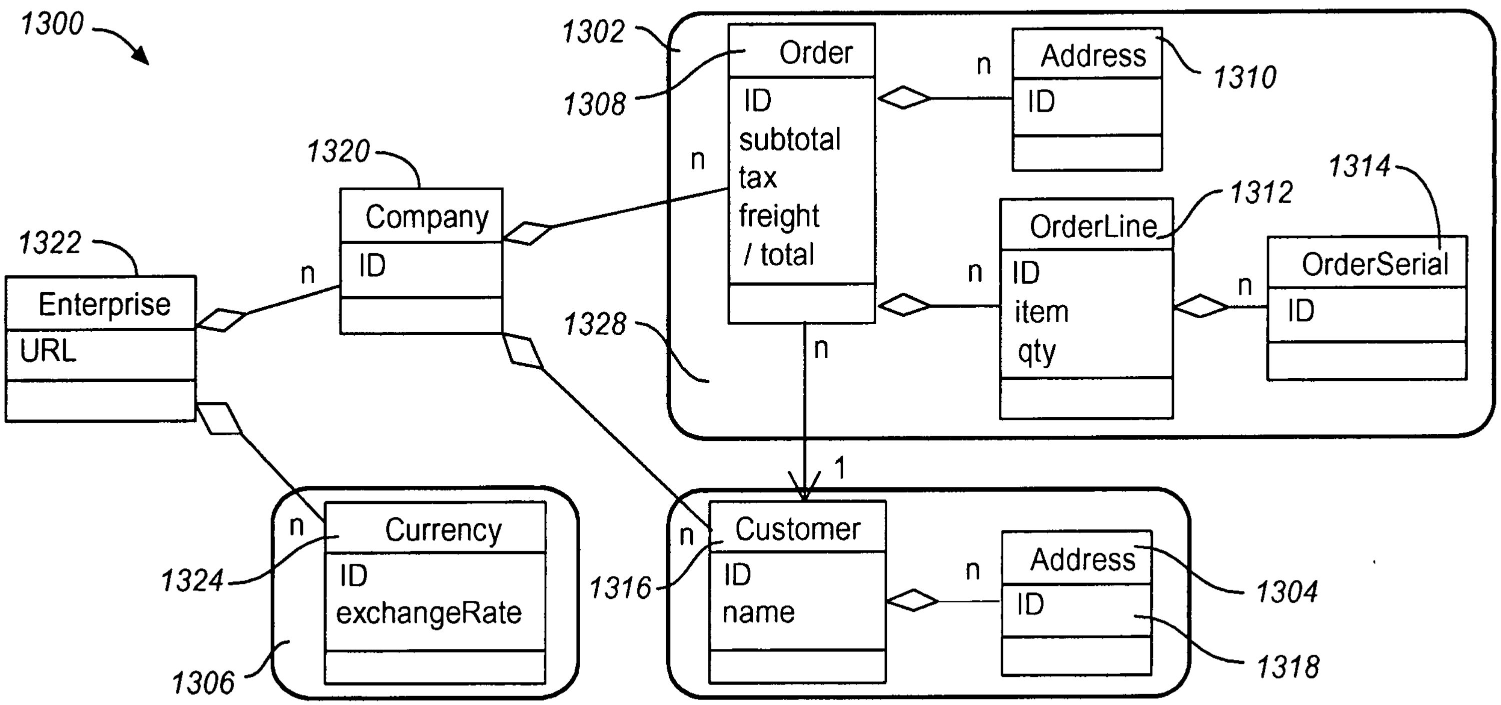

1300
1302
1308
Order
ID
subtotal
tax
freight
/ total
n
Address
ID
1310
n
1320
Company
ID
n
1322
Enterprise
URL
n
1312
OrderLine
ID
item
qty
n
1314
OrderSerial
ID
n
1328
n
1
Currency
ID
exchangeRate
n
1324
1306
Customer
ID
name
n
1316
Address
ID
n
1304
1318

FIG._ 28

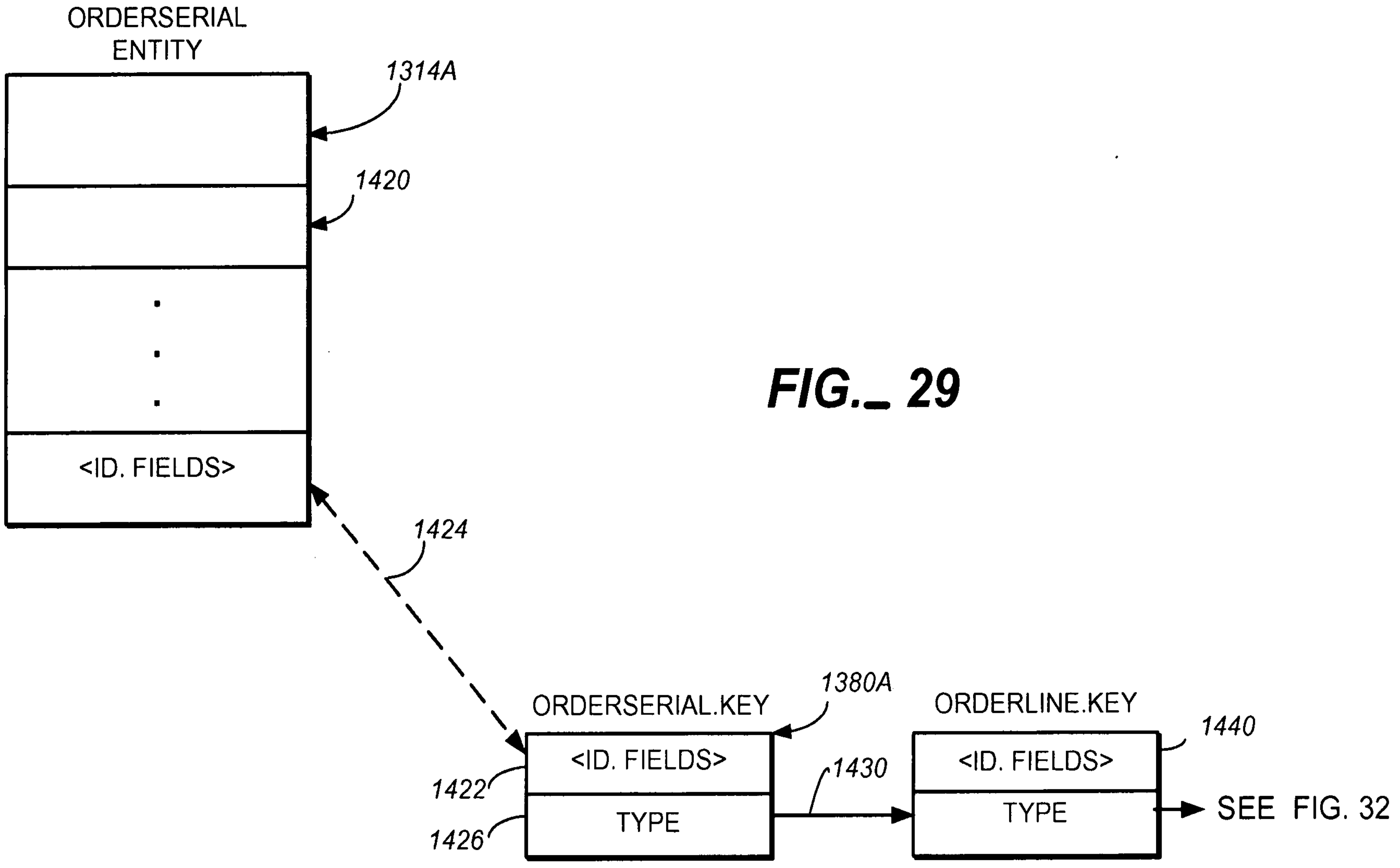

ORDERSERIAL ENTITY
1314A
1420
<ID. FIELDS>
1424
ORDERSERIAL.KEY
1380A
<ID. FIELDS>
TYPE
1422
1426
1430
ORDERLINE.KEY
1440
<ID. FIELDS>
TYPE
SEE FIG. 32

FIG._ 29

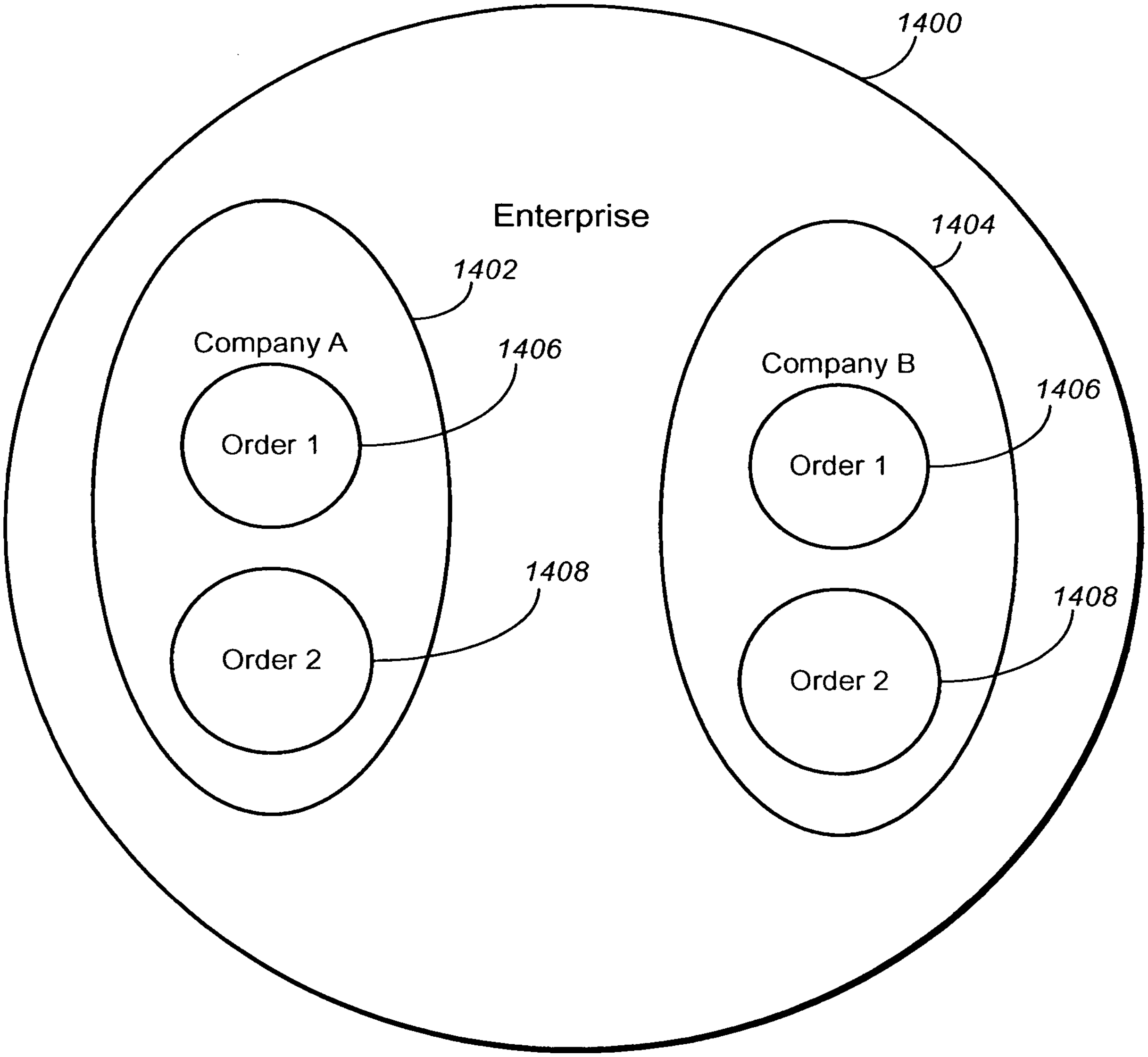

1400
Enterprise
1402
1404
Company A
1406
Order 1
Order 2
1408
Company B
1406
Order 1
1408
Order 2

FIG._ 30

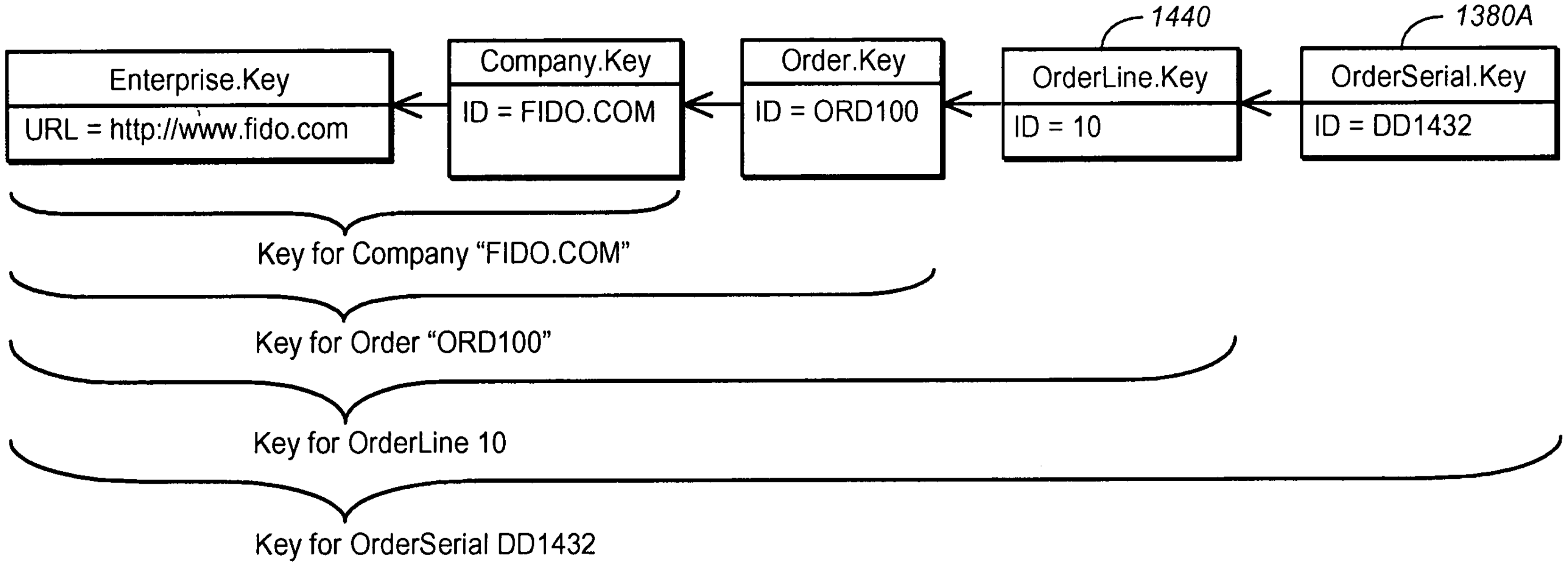

1440
1380A
Enterprise.Key
URL = http://www.fido.com
Company.Key
ID = FIDO.COM
Order.Key
ID = ORD100
OrderLine.Key
ID = 10
OrderSerial.Key
ID = DD1432
Key for Company "FIDO.COM"
Key for Order "ORD100"
Key for OrderLine 10
Key for OrderSerial DD1432

FIG._ 31

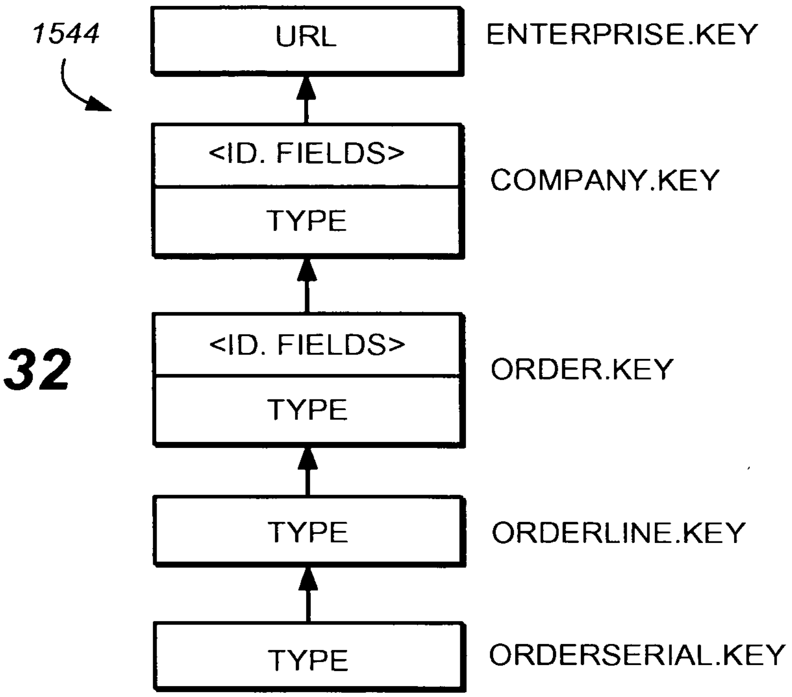


FIG._ 32

1350

COMPANY_ID 1352	ORDER_ID 1354	ORDERLINE_ID 1356	SERIAL NO. 1358	OTHER COLUMNS	
⋮	⋮	⋮	⋮	⋮	⋮

FIG._ 33